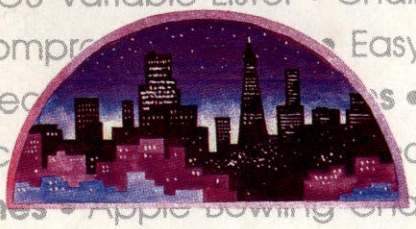
Inside on disk: The 44 best Apple programs ever published by COMPUTE!

COMPUTE!'s Best of APPLE

ProDOS Disk Inside

$12.95
1988
02714
$17.95 Canada
Display until
March 11, 1988

Includes **SpeedScript** A full-featured word processor



85
0 71486 02714 0

NEW! from Micro League Sports

MICRO LEAGUE Wrestling®

REAL WWF STARS FEATURING HULK HOGAN!

- ★ *Live, bone-crushing matches!*
- ★ *Your strategy—with authentic moves!*
- ★ *Ringside commentary, interviews and more!*
- ★ *Defend The Hulkster's title —or be the bad guy and try to take his title away!*

EXCLUSIVELY ENDORSED BY WWF WORLD WRESTLING FEDERATION

From the team who brought you Micro League Baseball—the exclusive, award-winning game simulation—another exclusive . . .

Micro League and World Wrestling Federation Match Disk!!

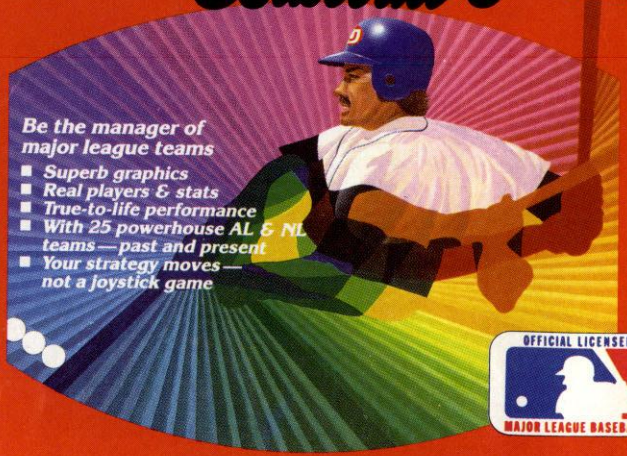
Featuring the Hulkster and some of the WWF's most infamous villains—all in real-life, digitized action.

Strap on Hulk Hogan's Championship belt or put on the bad guy's boots and *you* make the moves; body slam, leg drop, all the rest!

Take the challenge and step into the squared circle or pace in the dugout. Either way you'll be making all the strategy decisions.

Wrestling Disks available for: Commodore 64/128, Atari ST
Baseball Disks available for: Apple II Series, Commodore 64/128, IBM PC/PC Jr. and compatibles, Atari 65XE/130XE/800/XL.

MICRO LEAGUE Baseball®

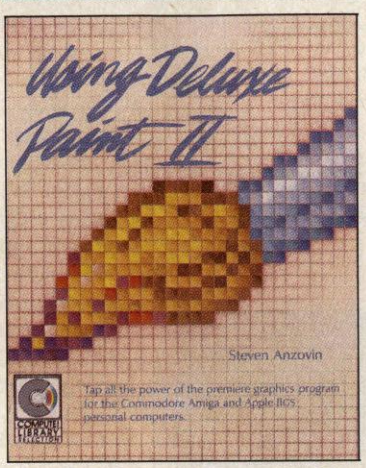


Micro League Sports Association
2201 Drummond Plaza, Newark DE 19711
302-368-9990

Check your software dealer or . . . direct Visa/MC orders call toll free 1-800-PLAYBAL

New Apple IIGS Books from COMPUTE!

Easy-to-use tutorials and ready-to-type-in programs show you how to get more from your Apple IIGS—from booting up the machine and programming in machine language to utilizing the advanced sound and graphics capabilities. Written in COMPUTE!'s clear, concise style, these books contain all the information you need to access the power of your IIGS.



Using *Deluxe Paint II*

Steven Anzovin
ISBN 0-87455-111-0 $18.95

Create the most spectacular art possible on a personal computer with this comprehensive, yet understandable, guide to *Deluxe Paint II*, the state-of-the-art graphics program for the Commodore Amiga and Apple IIGS. Simple drawing skills, customizing brushes, sophisticated palettes, special color effects, tools, styles, patterning, and perspective are just some of the techniques illustrated. Learn how to use every feature of *Deluxe Paint II*, from merging foreground and background to mixing graphics and text. A COMPUTE! Library Selection.

COMPUTE!'s Guide to Sound and Graphics on the Apple IIGS

William B. Sanders
ISBN 0-87455-096-3 $16.95

The impressive new machine from Apple holds the promise of dazzling graphics and symphonic sound—if you know how to use them. This tutorial, by the author of *The Elementary Apple IIGS*, begins by showing how to create graphics and sound using Applesoft BASIC, but it doesn't stop there. The book includes a multitude of programs, routines, and utilities with which you can open the IIGS's Toolbox, a sophisticated set of programming tools, so that you can produce truly amazing sights and sounds. A COMPUTE! Library Selection.

The Elementary Apple IIGS

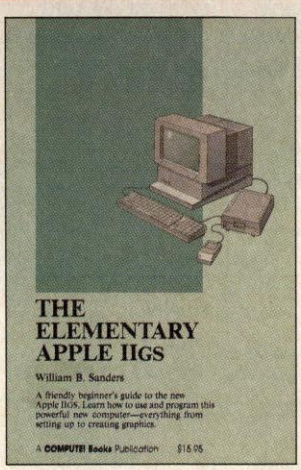
William B. Sanders
ISBN 0-87455-072-6 $15.95

A friendly, easy-to-use guide to the newest Apple computer, this book leads you through the steps of connecting the computer, loading programs, creating graphics, and writing programs. For both novice and seasoned programmers, it's an introductory text for everyone.

COMPUTE!'s Apple IIGS Machine Language for Beginners

Roger Wagner
ISBN 0-87455-097-1 $19.95

The latest in a series of introductory machine language books, *COMPUTE!'s Apple IIGS Machine Language for Beginners* is a clear and concise tutorial to learning the IIGS's native language. Written by noted Apple columnist Roger Wagner, this book includes many programming examples and clear explanations that make learning 65816 machine language easy. For beginning and intermediate machine language programmers as well as those who know another machine language and want to move up to this fast 16-bit language. A COMPUTE! Library Selection.



COMPUTE! Publications, Inc. abc
Part of ABC Consumer Magazines, Inc.
One of the ABC Publishing Companies

COMPUTE! books are available outside the United States from subsidiaries of McGraw-Hill International Book Company.

Look for these COMPUTE! books at your book or computer store. You can order direct by calling toll free 1-800-346-6767 (in NY 212-887-8525) or by writing COMPUTE! Books, Customer Service, P.O. Box 5038, F.D.R. Station, New York, NY 10150. Customer Service hours are 10:00 a.m.–12:30 p.m. and 1:30 p.m.–3:00 p.m. EST.

Please include $2.00 per book for U.S. or surface mail or $5.00 for airmail.

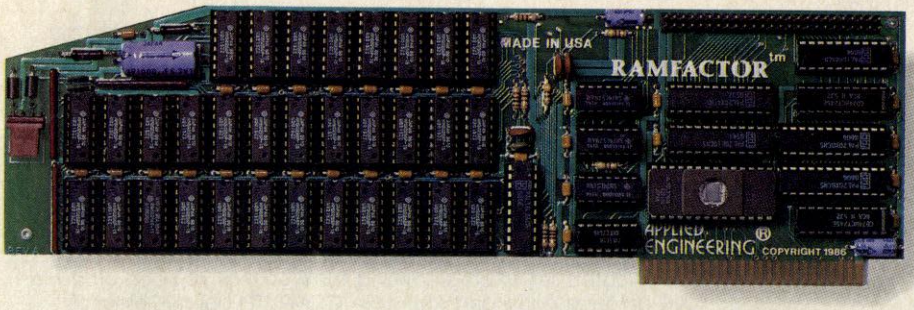
NC residents please add 5 percent sales tax and NY residents add 8.25 percent sales tax. Please allow 4–6 weeks for delivery.

RamFactor™

Boot any II+, IIe, or IIGS and be up and running your favorite software in less than 1 second.

Now anyone with an Apple II+, Franklin, Laser 128, Apple IIe or IIGS preferring to use slots 1 through 7 can now enjoy the speed and performance that they've been waiting for.

With RamFactor, you'll be able to instantly add another 256K, 512K, or a full 1 MEG on the main board and an expansion connector can address an additional 16 MEG of memory for possible future expansion. Since virtually all software is automatically compatible with RamFactor, you'll immediately be able to load programs into RamFactor for instantaneous access to information. You'll also be able to store more data for larger word processing documents, bigger data bases, and expanded spreadsheets. And with the battery back-up option, you can switch on your Apple and your favorite software will be up and running in less than 1 second!



Very Compatible

All the leading software is already compatible with RamFactor. Programs like AppleWorks, Pinpoint, BPI, Managing Your Money, Dollars and Sense, SuperCalc 3A, PFS, MouseWrite, MouseDesk, MouseCalc, Sensible Speller, Applewriter IIe, Business Works, ReportWorks, Catalyst 3.0 and more. And RamFactor is fully ProDos, DOS 3.3, Pascal 1.3 and CP/M compatible. In fact, no other slot 1-7 memory card is more compatible with commercial software.

AppleWorks Power

There are other slot 1-7 cards that give AppleWorks a larger desktop, but that's the end of their story. But RamFactor is the only slot 1-7 card that increases AppleWorks internal memory limits, increasing the maximum number of records in the database and lines permitted in the word processor, and RamFactor is the only standard slot card that will automatically load all of AppleWorks into RAM dramatically increasing speed and eliminating the time required to access the program disk, it will even display the time and date on the AppleWorks screen with any ProDos clock. RamFactor will automatically segment large files so they can be saved on 5¼", 3½", and hard disks. All this performance is available on the Apple II+, IIe, Franklin, or Laser 128 when used with an 80 column display. RamFactor, no other standard slot card comes close to enhancing AppleWorks so much.

The "Instant On" Apple.

With the RamCharger battery backup option, RamFactor will retain programs and data during power failures or even when you turn your computer off. And because RamFactor plugs into slots 1-7, it's the only memory card that can cold boot an Apple without using slow floppy or hard disks. An available "Y" cable allows one RamCharger to power two RamFactors.

Powerful Program Switcher

With RamFactor, you can organize memory into multiple work areas and switch between them. Each work area can contain different programs and even different operating systems. Now you can switch from one program to another or even switch from AppleWorks to DOS 3.3 to CP/M to Pascal to ProDos in under a second. And with our battery back-up option, you can have permanent storage for up to 20 years.

Features:

- 256K to 1 MEG on main board. Expansion connector for possible further expansion to 16 MEG.
- Fully Apple II Memory Expansion compatible
- Compatible with Apple IIe, II+, IIGS, Franklin and Laser 128
- Battery back-up option allows you to turn on your Apple and run your favorite programs in less than 1 second!
- Automatically recognized by ProDos, DOS 3.3, Pascal, and standard CP/M
- Built-in RamDrive™ software (a true RAM disk not disk caching)
- Systems are directly bootable from RamFactor if desired
- Built-in self diagnostic software
- Automatic expansion with AppleWorks 1.3 or 2.0
- Allows Apple II+ and IIe to run your AppleWorks without buying any additional software
- Accelerates AppleWorks
- Displays time and date on the AppleWorks screen with any ProDos clock
- Fits any I/O slot except slot 3
- Fully socketed and user upgradeable
- Much, much more

RamFactor with 256K	**$239**
RamFactor with 512K	**$279**
RamFactor with 1 MEG	**$359**
RamFactor with 2-16 MEG	**CALL**
RamCharger backup option	**$179**
"Y" cable	**$24**

(Allows one RamCharger to power two RamFactors.)

Order RamFactor today . . . with 15 day money back guarantee and our "no hassle" five year warranty. *See* your dealer *or call* (214) 241-6060, 9 a.m. to 11 p.m., 7 days, *or send* check or money order to Applied Engineering, MasterCard, Visa and C.O.D. welcome. Texas residents add 6¼ sales tax. Add $10.00 if outside U.S.A.

Applied Engineering™

The Apple enhancement experts.

(214) 241-6060

P.O. Box 798, Carrollton, TX 75006

COMPUTE!'s Best Of APPLE

GAMES

PROGRAMMING

GRAPHICS

APPLICATIONS

PROGRAMMING

Publisher	James A. Casella
Editorial Director	Richard Mansfield
Managing Editor	Kathleen Martinek
Associate Publisher	Selby Bateman
Production Director	Tony Roberts
Editor, COMPUTE! and COMPUTE!'s GAZETTE	Lance Elko
Editor, COMPUTE!'s Atari ST Disk & Magazine and COMPUTE's PC Magazine	Tom R. Halfhill
Editor, COMPUTE's Apple Applications Magazine	Gregg Keizer
Features Editor	Keith Ferrell
Technical Editor	Ottis R. Cowper
Assistant Technical Editors	Dale McBane, Jim Fuchs
Assistant Editors	Todd Heimarck, Rhett Anderson, John Shadle, Randy Thompson, Cliff Karnes
Programming Supervisor	Patrick Parrish
Editorial Programmers	Tim Victor, Tim Midkiff, William Chin, George Miller
Buyer's Guide Coordinator	Caroline Hanlon
Copy Editors	Karen Uhlendorf, Karen Siepak, Jill Champion
Submissions Reviewer	David Hensley
Programming Assistants	Troy Tucker, Joyce Sides
Executive Assistant	Debi Nash
Administrative Assistants	Julia Fleming, Iris Brooks, Sybil Agee
Receptionist	Anita Armfield
COMPUTE!'s Book Division Editor	Stephen Levy
Assistant Editors	Tammie Taylor, Robert Bixby, Lynne Weatherman
Programming Assistant	David Florance
Director of National Sales	Joseph W. Hatcher
Production Manager	Irma Swain
Art Director	Janice R. Fary
Assistant Art Director	Lee Noel, Jr.
Assistant Production Manager	De Potter
Artists	Robin Case, Kim Potts, Scotty Billings, Tony Jacobson
Typesetting	Terry Cash, Carole Dunton
Illustrator	Harry Blair
Director of Advertising Sales	Peter Johnsmeyer
Advertising Director	Bernard J. Theobald, Jr.
Production Coordinator	Kathleen Hanlon
Customer Service Manager	Diane Longo
Dealer Sales Supervisor	Jose Cruz
Individual Order Supervisor	Cassandra Green

James A. Casella, President
Richard Mansfield, Vice President, Editorial Director
Richard J. Marino, Vice President, Advertising Sales
Ilene Berson Weiner, Vice President, Production

Editorial Board
Richard Mansfield, Kathleen Martinek, Selby Bateman, Lance Elko, Tom R. Halfhill, Stephen Levy
Robert Lock, Founder and Editorial Consultant

Editorial offices:	324 West Wendover Avenue Suite 200 Greensboro, NC 27408 USA
Corporate offices:	825 7th Avenue New York, NY 10019 212-265-8360
Customer Service:	800-346-6767 (In NY 212-887-8525) 10:00 AM–12:30 PM; 1:30 PM–3:00 PM Monday–Friday
Dealer Sales:	800-638-3822 (In NY 212-887-8566) 9 AM–5 PM Monday–Friday

Advertising Sales Representatives

New England & Mid Atlantic
Bernard Theobald, Thomas Link: 212-315-1665

Midwest & Southwest
Jerry Thompson, Lucille Dennis: 312-726-6047 (Chicago), 713-731-2605 (Texas), 303-595-9299 (Colorado), 415-348-8222 (California)

West, Northwest & British Columbia
Jerry Thompson, Lucille Dennis: 415-348-8222

Southeast & International
Harry Blair: 919-275-9809

Send all advertising materials to:
Kathleen Hanlon
324 West Wendover Ave., Suite 200
Greensboro, NC 27408

Editor's Notes

When personal computers were still considered new inventions—way back about ten years ago—there was virtually no commercial software market. Even five years ago, almost all commercial software companies were relatively small concerns, unlike the multimillion-dollar firms that dominate the business today. Thus were born computer magazines that published program listings to fill the software void.

Computer magazines remain an important source of software to this day. For a few dollars, they offer solid, useful programs and serve as an outlet for talented programmers. In nearly every user's software library are favorite utilities, applications, educational programs, and games that originated in a magazine.

COMPUTE! magazine has published many outstanding programs over the past eight years. Recently we decided to collect the best of these programs into special magazine/disk editions such as the one you're reading now. Almost all of the back issues of COMPUTE! are out of print, and thousands of new users will find this collection invaluable.

Collected in this special issue are the best home and recreational programs published by COMPUTE! for the Apple II line of computers over the past several years. There's something here for every age, every level. The very young can learn to tell time with "Hickory Dickory Dock"; older children can study the stars and planets with "Skyscape"; programmers can see how the computer stores information on disk with the valuable "Dr. Disk" utility.

For game players, we've included action games and thinking games. Among them you'll find an exceptional version of the traditional game of "Chess," as well as simulation of Backgammon. This issue also includes *SpeedScript*, a full-featured word processor, ready to load and run.

Using The Programs

Packed with this magazine is a ProDOS-formatted disk containing all of the programs and examples described in these pages. Program files are stored on both sides of the disk; if you don't find the files you need, turn the disk over and try again. These programs are ready to run and use—there's no need to type in and debug long printed listings. Before you begin, however, we recommend that you make a copy of this disk and then store the original safely away as a backup. The disk is not copy-protected. It is, however, write-protected — you can't store additional data or modify the programs *on this disk.* Since some of the programs in this collection are designed to write to the disk, you *must* copy these programs to another disk for them to work properly.

Before using a program, check the introductory paragraph of the article which describes the program to see what disk files are used and what hardware is required. Then follow the instructions in the article for loading and running the program.

Most of these programs can be used without changes under either ProDOS or DOS 3.3. A few programs, *SpeedScript*, for example, have separate versions to be used if the programs are to be run under DOS 3.3. The articles clearly indicate when a special version is required for DOS 3.3, and the filenames for the DOS 3.3 versions all include the characters "3.3".

We're certain you'll enjoy these programs as much as the regular readers of COMPUTE! have over the years, and we invite you to look for COMPUTE! and the COMPUTE!'s Apple Applications Magazine for valuable information and software for your Apple in the coming years.

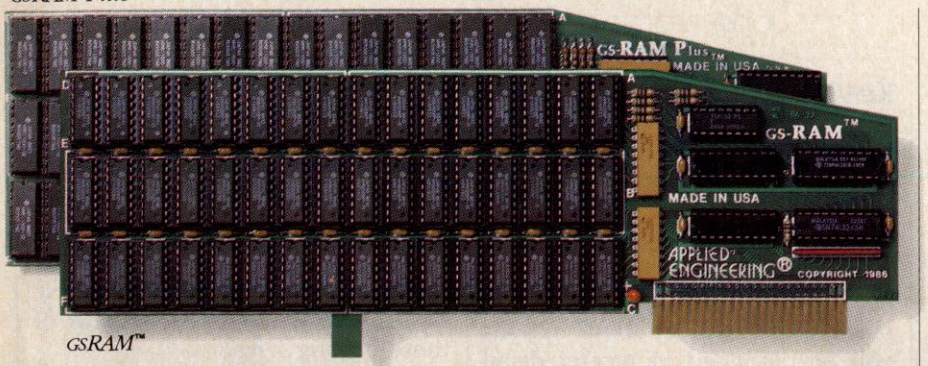
COMPUTE! Publications, Inc.
Part of ABC Consumer Magazines, Inc.
One of the ABC Publishing Companies
ABC Publishing, President, Robert G. Burton
1330 Avenue of the Americas, New York, New York 10019

COMPUTE!'s Best of Apple is published by COMPUTE! Publications, Inc., 825 7th Ave., New York, NY 10019 USA. Phone: (212) 265-8360. Editorial Offices are located at 324 West Wendover Avenue, Greensboro, NC 27408.

Insist on GSRAM™ When You Buy Your IIGS™

Expand the IIGS RAM and ROM with the GSRAM or GSRAM Plus with ROM Pak. Available now with 256K to 8 MEG!

GSRAM Plus™



GSRAM™

Remember the 16K cards for the II+ and the 64K cards for the IIe? At the time, that much memory seemed like a lot. But when the owners of these memory cards came to us for more memory, many had to throw away their smaller Apple memory cards or try to sell them. Most of our customers told us that had they known about Applied Engineering's larger memory cards when they bought their Apple, they would have purchased them at the same time.

GSRAM and GSRAM Plus are available now, allowing up to 8 MEG of memory expansion. That's 8 times the memory capacity of Apple's card and just look at the benefits that only GSRAM and GSRAM Plus have over Apple's card:

- Lower cost
- Has 6 RAM banks (Apple's card has 4)
- Has memory expansion port
- Has ROM expansion port
- No configuration blocks to set
- No soldered in RAM chips
- Expandable to 8 MEG
- Expands AppleWorks internal limits
- Built-in Hi-Res self-diagnostic software
- 5 year hassle free warranty (Apple has a 90 day warranty)
- Made in USA

GSRAM for More AppleWorks Power

Only GSRAM and GSRAM Plus eliminate AppleWorks internal memory limits, increasing the maximum number of records available from 6,000 to over 25,000 and only GS-RAM and GSRAM Plus increase the number of lines permitted in the word processing mode from 6,000 to over 15,000. And only GSRAM and GSRAM Plus offer a built-in printer buffer so you can continue using Appleworks while your printer is printing. GS-RAM and GSRAM Plus even expand the number of lines in the dipboard from 255 to 2047 and will auto segment large files so they can be saved on two or more disks. You can even have Pinpoint or Macroworks and your favorite spelling checker in RAM for instant response. GSRAM and GSRAM Plus will even display the time and date right on the AppleWorks screen. Nothing comes close to enhancing AppleWorks so much.

Turn Your IIGS into a Giant

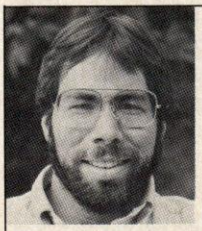
Simply plug GSRAM into the IIGS memory expansion slot and you've got up to *8 megabytes* of RAM at your fingertips—all of it instantly and automatically recognized by the IIGS. GSRAM is compatible with all IIGS software, including AppleWorks, as well as BASIC®, ProDos, DOS 3.3, PASCAL®, "C" and CP/M®.

Grow by Bytes or Megabytes

We offer GSRAM in two configurations so you can increase your memory 256K at a time (GSRAM) or a megabyte at a time (GS-RAM Plus). Both offer full compatibility, lower cost than other boards, and easy expandability. And both are extremely low in power consumption. A fully expanded GSRAM operates at only 375 ma, and GSRAM Plus at only 270 ma (even with 6 megabytes on board!)

GSRAM—for Normal Memory Requirements

GSRAM is available with 256K, 512K, 1 MEG or 1.5 MEG of memory already on board. If you don't need the full 1.5 MEG now, you can choose a GSRAM with less memory and expand it up to 1.5 MEG in the future—or upgrade to GSRAM Plus for a small charge.



"In quality, performance, compatibility, expandability and support, Applied Engineering's GSRAM and GSRAM Plus are number one."

Steve Wozniak, the creator of Apple Computer

With an optional piggyback card, you can expand GSRAM even higher than 1.5 MEG! (Other cards are only expandable to 1 MEG.)

GSRAM Plus—for Growing by Leaps and Bounds

GSRAM Plus is the first Apple memory card to use 1 MEG RAM chips on the main board. It's available with 1 to 6 MEG on board. If you don't need the whole 6 MEG now, you can buy a GSRAM Plus with less memory and easily expand it in the future.

GSRAM Plus can be expanded up to 8 MEG with an optional piggyback card.

Easy Expansion

Both GSRAM and GSRAM Plus use standard RAM chips that are readily available and just plug right in. So unlike other cards, you'll find expanding your GSRAM or GSRAM Plus easy, convenient and very economical. And with our optional ROM expansion module you can even increase the IIGS's ROM space and all in just one slot.

Quality and Support of the Industry Leader

Applied Engineering is the largest supplier of Apple peripherals in the world. We *invented* the first large RAM cards for the Apple. With a 5-year "no-hassle" warranty and outstanding technical support, you can be sure GSRAM and GSRAM Plus will deliver the performance you're looking for—or return them within 15 days for a full refund.

GSRAM with 256K	$169
GSRAM with 512K	$219
GSRAM with 1 MEG	$299
GSRAM with 1.5 MEG	$379
GSRAM with 2-8 MEG	CALL
GSRAM PLUS with 1 MEG	$459
GSRAM PLUS with 2 MEG	$759
GSRAM PLUS with 3-8 MEG	CALL

Order today!

See your dealer or call Applied Engineering today, 9 a.m. to 11 p.m. 7 days. Or send check or money order to Applied Engineering. MasterCard, VISA and C.O.D. welcome. Texas residents add 6¼% sales tax. Add $10.00 outside U.S.A.

APPLIED ENGINEERING™

The Apple enhancement experts.

(214) 241-6060

P.O. Box 798, Carrollton, TX 75006

CP/M is a registered trademark of Digital Research, Inc.

Chess

John Krause

This impressive chess game is not only fast, but includes five levels of play. Play the computer, let the computer play itself, or use the game to solve difficult chess problems. For Apple II+, IIe, and IIc computers with at least 48K RAM and a disk drive. Disk File: CHESS.

The world was amazed, in the late eighteenth century, by a machine that had the astonishing ability to play a good game of chess. It entertained kings and queens. It defeated Napoleon, a master tactician. Hundreds of people paid to compete against it. Eventually, however, it was revealed that a small man was hidden inside the machine.

A chess-playing machine remained only a dream until the late 1950s when the first computer chess game was played. Now, the World Computer Championship, held every three years since 1974, attracts almost as much publicity as the human championship matches. Why has there been so much interest in machines that play games?

One reason is that chess can be used to measure a computer's intelligence. Chess is easy to play, but difficult to master. So difficult, in fact, that some experts believe that a computer would have to be almost as intelligent as a human to become world champion.

Of course, another reason is that chess is just plain fun—but not if you can't find an opponent. To be an entertaining opponent, a computer chess game should be fast, easy to use, and capable of playing at several different skill levels. "Chess" has all these features, and more. Although it's really no match against the best commercial chess games, it *has* managed to defeat these giants of the microcomputer chess world on rare occasions.

Keyboard Input

After running the program, you'll be asked to specify several play options. You can choose among five skill levels; start a new game or set up any position; play against the computer or watch it play against itself; or play either the white or black pieces. All of these options will be discussed in greater detail later, but for now, just type 1 at each prompt. This puts you in command of the white pieces versus the computer on level 1, the easiest level.

The first time the program is run, you need to wait a few seconds while the computer gets its brain in order. Then the board will be displayed with your pieces on the bottom of the screen and the computer's pieces on the top. The square in the lower-left corner of the board should be blinking. This is the cursor which takes the place of your hand to move pieces around the board.

Use the A, D, W, and S keys to move the blinking square left, right, up, or down. Place the cursor square atop the piece you wish to move and press RETURN. Now maneuver the cursor to the square you want to move the piece to and hit RETURN again. Your piece appears in the new position, and the computer responds almost instantly with its move.

A Spectacular Blunder

Did you make a foolish move? No problem. One of the most valuable features of Chess is its ability to change piece positions by adding or deleting pieces. This feature is especially useful for those of us who frequently manage to maneuver into a superior position, only to throw it all away in a single, spectacular blunder.

A piece can be deleted by positioning the cursor on it and pressing the space bar. To add a piece or change a piece to a different one, move the cursor to the appropriate square and press P, N, B, R, Q, or K for pawn, knight, bishop, rook, queen, or king, respectively. This puts one of *your* pieces on the square. To add one of the computer's pieces, hold down the CONTROL key while pressing one of these editing keys.

To take back a move, use the editing keys to delete your piece and put it back on its original square. Don't forget to take back the computer's move, too.

The editing feature also enables you to make special moves which cannot be made with the keyboard alone, such as castling and *en passant* captures. For example, castling can be accomplished by deleting the king and putting it on its new square, and then moving the rook as you normally would with the keys. Although *you* can make these special moves, the computer will never castle or capture *en passant* because, due to their complexity, these moves were not included in its thinking routine.

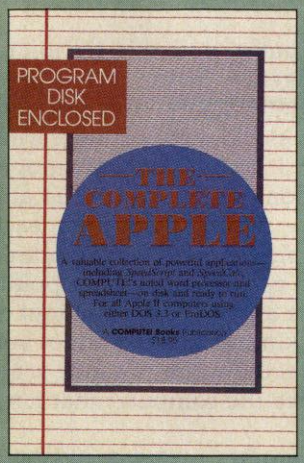
Strange Chess

Although the computer always makes a legal move, it doesn't check to see that you do the same. You're free to move any of your pieces to any square without so much as a contemptuous beep from the computer. If you're an experienced player, this shouldn't be a problem. If you're a beginner, however, you may want to familiarize yourself with the basic rules of chess lest you end up playing strange chess, a personal version which bears little resemblance to the real thing. On the other hand, if you like to fudge a bit, the computer makes it easy. It politely acquiesces to your most surreal moves.

When a pawn reaches the other side of the board, it's automati-

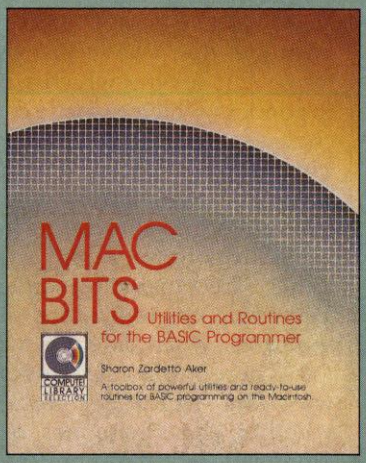
NEW COMPUTE! BOOKS FOR YOUR APPLE II AND MACINTOSH

Look to COMPUTE! Books for the best, most up-to-date information for your Apple II and Macintosh computers. You'll find clear guides to many commercial programs, tutorials to help you design your own programs, ideas for setting up a desktop publishing system, and powerful applications software.



COMPUTE!'s Quick and Easy Guide to *AppleWorks*™
Tom and Ellen Dougher
ISBN 0-87455-109-9 $10.95
The unchallenged leader in personal productivity software for the Apple II series, *Appleworks* is an integrated package combining a word processor, database, and spreadsheet. Precisely because *AppleWorks* can do so much, it can prove complicated even for a regular user. *COMPUTE!'s Quick and Easy Guide to AppleWorks* solves that problem, giving you instant access to all the information you're likely to need on a day-to-day basis.

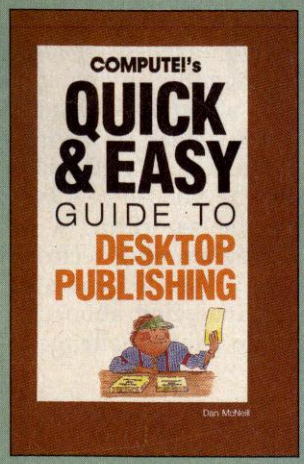
Clear, complete guidance will benefit first-time and experienced users alike. You'll see how to get started with *AppleWorks*, and exactly how to get the most out of each of the three major applications. You'll quickly master the intricacies of this powerful program using the special Command Summary, Keyboard Template, and Quick-Reference Chart.



Making Dollars and Cents with *Dollars and Sense*™
Gia Rozells and James Bartimo
ISBN 0-87455-101-3 $18.95
Take charge of your money with this guide to Monogram's *Dollars and Sense*, the bestselling personal and business financial-management program for the IBM PC and the Apple Macintosh. *Making Dollars and Cents with Dollars and Sense* takes you past the manual to fully exploit this efficient, versatile software. Graph income and expense, analyze cash flow, and see your real financial picture so you can make more informed decisions.

COMPUTE!'s Quick and Easy Guide to Desktop Publishing
Dan McNeill
ISBN 0-87455-112-9 $10.95
Desktop publishing is one of the fastest growing personal computer applications, making room for itself in businesses large and small. With the right computer, software, and printer, you can turn out documents, flyers, brochures, and more—all in the highest quality available outside of a professional print shop. This easy-to-use reference guide explains the ins and outs of desktop publishing, shows what you can do with the variety of available software, and even helps you select your system. If you're tired of wading through long and technical manuals, this guide and tutorial can get you started today with desktop publishing.

MacBits: Utilities and Routines for the BASIC Programmer
Sharon Zardetto Aker
ISBN 0-87455-075-0 $18.95
Book/disk combination
ISBN 0-87455-076-9 $29.95
A collection of valuable utilities and routines for the Macintosh with Microsoft BASIC. Utilities create windows, menus, and much more. Over 80 routines for functions such as refreshing the screen, scanning the keyboard, and keeping time. Information included on using the new libraries to enhance BASIC. *A disk is available which includes all the programs and most of the routines in the book, $15.95 (750BDSK).*

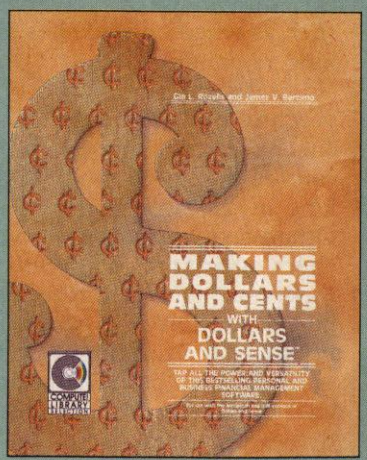


The Complete Apple
Edited
ISBN 0-87455-084-X Book/disk only $18.95
A book/disk combination with a series of powerful applications on disk. The book has complete, easy-to-follow documentation for each of the applications. Featured are *SpeedScript* and *SpeedCalc*, COMPUTE!'s professional-quality word processing and spreadsheet programs. All the applications are on disk ready to load and use on any Apple II computer.

COMPUTE! Books are at your local computer store or bookstore, or you can order directly by calling toll free 1-800-346-6767 (in NY 212-887-8525). Write COMPUTE! Books, Customer Service, P.O. Box 5038, F.D.R. Station, New York, NY 10150.

COMPUTE! books are available outside the United States from subsidiaries of McGraw-Hill International Book Company.

COMPUTE! Publications, Inc. abc
Part of ABC Consumer Magazines, Inc.
One of the ABC Publishing Companies



Please include $2.00 shipping and handling per book for U.S. and surface mail, $5.00 for airmail. NC residents add 5 percent sales tax, and NY residents add 8.25 percent sales tax. Please allow 4–6 weeks for delivery.

cally promoted to a queen. If you would rather have a knight, bishop, or rook, you can easily make the change using the editing keys.

Checkmate

The computer thinks by analyzing thousands of possible moves and countermoves and choosing what it considers to be the best move based on the relative value of the pieces (see "How 'Chess' Thinks"). Most positions don't have just one best move, but several which are equally good, in which case the computer chooses among them at random. This random factor insures that every game is different, and makes for varied and interesting play.

Play continues until one side is either checkmated or stalemated. The computer then stops play and indicates which side has won.

There are a few quirks in the way the computer determines whether checkmate has occurred. On levels 3–5, it announces checkmate prematurely. When this happens, the computer has determined that it's impossible to avoid checkmate on the *next* move or two, assuming both sides make the best moves.

Also, the computer doesn't know the subtle difference between checkmate and stalemate. Consequently, when stalemate occurs, it announces checkmate although, in fact, the game is a draw. Since the computer tries as hard as it can to checkmate its opponent, it also tries to achieve stalemate, possibly forcing a draw when it could have won. Fortunately, this rarely happens because the conditions for stalemate exist only in unusual circumstances such as when one side has only the king remaining.

Also, the computer won't give you any hint when your king is in check (not checkmate). So be extra careful that you don't leave your king in check or move into check. Otherwise, your king would be in check during the computer's turn to move—a highly unorthodox if not illegal position. The computer's reply to such a position is unpredictable, but it usually announces checkmate, forcing you to restart the game.

In any case, when the computer announces checkmate, press

How "Chess" Thinks

You've probably heard that if a monkey sat down at a typewriter and pecked randomly at the keys for a long enough period of time, it would eventually type the complete works of Shakespeare. Theoretically, that's possible—given enough time. There's the rub. At a brisk typing speed of 50 words per minute, it would take that poor monkey billions of years just to type "To be, or not to be." Nevertheless, there *is* power in trial and error.

The Minimax Algorithm

Substitute the monkey for a high-speed computer, and this technique becomes a practical method of imitating intelligence. In fact, it has been used with great success in the field of artificial intelligence. This program uses a popular trial-and-error technique known as the *minimax* algorithm.

The computer looks at the present board position and mentally moves the pieces through all the possible combinations of future moves and countermoves up to a certain point, say three moves ahead. For each combination, it calculates a score based on which pieces were captured during the combination. Each piece is worth a certain number of points depending on its general importance: 1 point for a pawn, 3 for a knight or bishop, 5 for a rook, 9 for a queen, and 46 for a king. (Of course, since you lose the game if your king cannot escape capture, the value of a king is actually infinite, but 46 is high enough to convince the computer that it's a bad move.)

When, in a move being examined, the computer captures an opponent's piece, the value of that piece is added to the score. Conversely, when one of the computer's pieces is captured, its value is subtracted from the score. Thus, a high score is considered good for the computer, a low score good for its opponent.

The task is to find the combination that represents best play for *both* sides. This combination is not necessarily the one with the maximum score, because while the computer is trying to maximize the score, its opponent is trying just as hard to minimize it. The best combination gives maximum scores during the computer's moves, minimum scores during the opponent's moves.

After the best combination has been found, the computer's best move in the present position is simply the first move in the combination. The problem has been reduced from analyzing a chess position to finding the maximum and minimum of a series of numbers, which is much better suited to a computer.

50 Million Combinations On Level 5

Like most algorithms based on trial and error, this one requires sifting through an enormous number of combinations to find the best one. Fortunately, a few tricks can be used to reduce the combinations to a manageable number. This algorithm uses a technique called *alpha-beta cutoff*. It makes the computer search more intelligently, giving it the seemingly paradoxical ability to find the best move without looking at all the possible combinations. On level 5, for example, instead of having to search through roughly 2 billion combinations, it looks at only 50 million.

Even so, it would take BASIC from now till the middle of 1986 to generate that many combinations. That's why the algorithm is programmed in machine language. An advanced programming technique known as *recursion* (making a subroutine call itself) is used to generate all the possible combinations of moves. Capable of analyzing about 5000 combinations per second, this routine provides a moderate challenge at a reasonable playing speed.

any key to start a new game. If you want to try out some of the other play options without waiting till checkmate, you can start a new game at any time by pressing CONTROL-RESET and rerunning the program.

Play Options

When you choose the black pieces, the board revolves so that you're still playing from the bottom. Since the player with the white pieces always moves first, you must wait for the computer to move before you can make your first move.

If you become mentally exhausted after several bouts against the computer, give your brain a rest and watch the computer play itself. When you select this option, just sit back and watch the action. Beginners will find this feature an excellent way to learn some good strategies to use against the computer.

You don't have to begin a game from the starting position. If you choose the option to set up a position, an empty board is displayed—you can place pieces in any position with the editing keys. When the position is set up, the computer starts thinking after you make your first move.

This feature is especially useful for continuing a previous game or creating a problem for the computer to solve. It also allows you to experiment with hypothetical or downright ridiculous positions. Live out your fantasy by giving yourself ten queens versus the computer's lone king. The position doesn't even have to be a legal one. You could invent your own type of chess by giving each side two kings, for example, although the computer may get confused trying to determine when checkmate has occurred.

One of the advantages of a computer opponent is that you can tell the machine exactly how hard you want it to try to beat you. It obediently plays at that level of difficulty. This is important, because it's no fun if you always lose or always win effortlessly.

You have five skill levels to choose from. The difference between one level and another is the number of moves ahead that the computer looks. On level 1, for example, it looks two moves ahead (its move and your reply). Each succeeding level looks ahead one more move than the previous level.

Alas, the smarter play on the higher levels doesn't come without a price. The further ahead the computer looks, the more moves it must examine and, hence, the longer it thinks. The thinking time varies greatly depending on the level (about one second per move on level 1; about two *hours* on level 5).

Here's a rundown of the five levels:

Level 1: Beginner. Thinking time: one second. Look ahead: two moves. Fast but dumb.

Level 2: Intermediate. Thinking time: five seconds. Look ahead: three moves. Provides a reasonable challenge for impatient players.

Level 3: Tournament. Thinking time: two minutes. Look ahead: four moves. Since the usual time limit for tournament play is 40 moves in two hours, an average of three minutes per move, this level is best suited for serious players.

Level 4: Mate in two. Thinking time: 30 minutes. Look ahead: five moves. Capable of solving most mate-in-two problems.

Level 5: Postal chess. Thinking time: two hours. Look ahead: six moves. Simulates postal chess games where there is no time limit. Can avoid checkmate in two moves.

The thinking times given here are average times. The actual time ranges from half to twice the average time, depending on the position.

Level 4 can be used to solve mate-in-two problems such as those published in many newspapers. Just select the following options: level 4, set up position, and computer versus itself. Enter the positions using the editing keys, and then make a do-nothing move by positioning the cursor over a white piece and pressing the RETURN key twice. After several minutes of deep thought, the computer should respond by moving one of the white pieces (the solution) and announcing checkmate. The only mate-in-two problems that the computer cannot solve are those which involve castling, *en passant* captures, or pawn promotion. ©

Space Caverns

Vince Valenti
Version by Rob Terrell

Hostile aliens track you down with the tenacity of bloodhounds in "Space Caverns," an action game for multiple computers. Requires a joystick. Disk File: SPACE.CAVERNS.

The object of "Space Caverns" is to elude four maniacal aliens who relentlessly home in on your diamond-shaped ship. You can move the ship up, down, left, and right, but there is no escape from the dangerous caverns—you can only advance further into the depths, shooting the aliens before they clobber you.

The aliens aren't dummies, either. They are fairly intelligent and will seek you out faster than you can run away. If you manage to defeat them in several successive rooms, you get a bonus round in which you stalk sedentary aliens that randomly appear around the screen. A countdown clock limits the amount of time you have to capture these aliens. When it runs out, you advance to a harder level within the caverns.

After each game, the three highest scores flash on the screen, challenging you to do better next time.

Plug in a joystick to control your ship and use the first fire button to activate the laser.

On higher levels, obstacles and a central cave appear on the screen, but you can shoot through them to defend yourself against the aliens.

During the bonus rounds, stationary aliens are worth 100 points each. ©

Lexitron

Ron Wilson
Version by Tim Victor, Editorial Programmer

Like a bowl of alphabet soup, the "Lexitron" screen appears to be just a jumbled mass of letters. Can you find the ten hidden words before time runs out? A challenging game with three skill levels for the Apple II-series computers in either DOS 3.3 or ProDOS. Disk File: LEXITRON.

If you enjoy the hidden word games often found in newspapers and magazines, you'll like "Lexitron." Unlike those you solve with pencil and paper, Lexitron includes a few twists. There's a time limit—and you can select one of three difficulty levels.

The game is written entirely in BASIC for the Apple II+, IIe, and IIc computers (running under either DOS 3.3 or ProDOS).

Side To Side, Top To Bottom

With Lexitron on disk, simply type:

RUN LEXITRON

After a few moments, you'll see the game's three different skill levels.

- Level 1—All the hidden words are spelled left to right, or top to bottom. No diagonals are used.
- Level 2—More difficult, it includes words formed in reverse order (from right to left or bottom to top), as well as forward-spelled words. Again, no diagonals are used.
- Level 3—The most difficult, with both forward and reversed diagonal words being formed. You might want to stay clear of level 3 until you've played a few times.

Once you've chosen a skill level by pressing the appropriate number key, hit Return. The screen will clear for a few seconds while the game words are being selected and hidden. Don't leave your seat—the timer starts as soon as the game appears on the screen.

Using the cursor keys, move the cursor to any word you've found and press the Return key while the cursor is on each letter in that word. You'll have to keep pressing one of the cursor keys, then the Return key, to complete the word. If you're entering characters in a hidden word, you'll see the message *Good Answer* at the bottom of the screen. Pressing the Return key on an incorrect letter gives you *Try Another.* When you've entered an entire hidden word, the words *Well Done* appear.

If you have an Apple II+—which has only left and right cursor keys—use the following Control key combinations to move the cursor up and down.

Up: **Ctrl-K**
Down: **Ctrl-J**

Notice that the cursor *wraps* around the screen. In other words, if the cursor is on the top line and you press the cursor up key, the cursor appears on the bottom line of the game board. Ditto for the left and right edges. This can speed things up when you're rushing to get to that last word and the time is almost up.

Each time a correct letter is registered, the time level, which moves from top to bottom, is pushed back toward the top. Avoid guessing letters by trial and error. Wrong entries only reduce the amount of time. When the time reaches the bottom of the screen, the game ends. All the hidden words are highlighted and you'll have the option to play another game, or stop.

All valid game words are at least six letters long. This keeps the accidental (and sometimes humorous) letter combinations from taking your time and effort. You'll often see words like MAN, CAR, SEE, or TRY, but Lexitron doesn't recognize them. Also be aware of letter additions. For example, Lexitron may choose and hide the word AMERICA, but by sheer chance the letter following could be an N, making AMERICAN. Lexitron may not recognize the extra N.

Easy Changes

The Lexitron vocabulary words are coded so that players can't list the program and get an illegal sneak preview. If you want to add your own words, or even entirely replace those in the program, the code is simple. Each letter represents the letter which alphabetically follows. For example, the letter A is coded as B. *ABACUS* would be coded as *BCBDVT.* If you decide to add your own words, start with a new line—1220—and remember that all words must be at least six letters long. *Be sure the last word in the list is FOE* (END in code). This signals the program that it's reached the end of the word list. You might want to avoid using words with the letters X, Z, or the Q-U combination. A sharp player can spot words with those letters in seconds. By the way, if you *do* want to use the letter Z, enter a left bracket character ([) as code in the DATA statement.

Lexitron, as it's written, can have a total of 200 words (not counting FOE) in its list. However, it will randomly select only ten from that list to include on the board.

If you find Lexitron too easy or too difficult, you can change the value *.009* in line 330. This controls the timer. Raising and lowering this value changes the allotted time, and thus the difficulty of the game. A value less than .009 (such as .007) makes the game easier, and, conversely, increasing the value makes it more difficult.

There are a few strategies to consider when playing Lexitron. For instance, in some cases it's not to your best advantage to enter a word as soon as you find it. If you have trouble finding some of the hidden words, Lexitron randomly provides a clue by flashing a word at the bottom of the screen. ©

Switchbox

Todd Heimarck, Assistant Editor
Version By Tim Victor, Editorial Programmer

Here's a challenging game of strategy that looks easy at first, but takes time to master and permits many variations. Disk File: SWITCHBOX.

Playing "Switchbox" is like putting dominos in place for a chain reaction—either you're setting them in position or you're knocking them over. Winning requires skill and a sense of when to go for points and when to lay back and wait for a better board. The goal is simple: You try to score more points than your opponent by dropping balls into a box full of two-way switches. Each switch has a trigger and a platform. If the ball lands on an empty platform, it stops dead. But if it hits a trigger, it reverses the switch and continues. In many cases dropping a single ball creates a cascading effect—one ball sets another in motion, which sets others in motion, etc., all the way down.

A Box Of Switches

Switchbox is a tale of twos: Each switch has two parts, two positions, two states, two paths in, and two paths out. The two parts are the platform and the trigger. A switch can lean to the left (platform left, trigger right) or to the right (platform right, trigger left):

The trigger is weak, and always allows balls to pass. But the platform is strong enough to hold a single ball. So the platform either holds a ball—it's full—or it does not and is empty. When a ball sits on a platform, the switch is said to be loaded, or full.

Figure 1: Trigger States

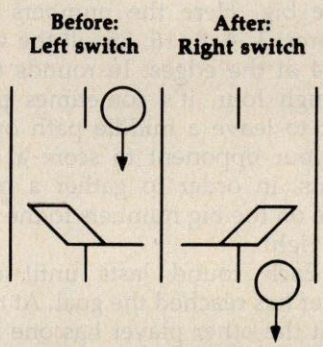


Figure 2: Loaded Trigger

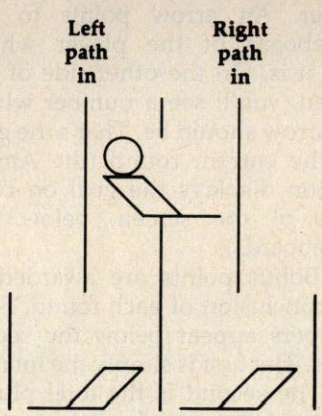


Figure 2 shows a full switch over two empty switches. The platform holds a ball and leans to the left. The trigger extends to the right. Note that the switch on top has two pathways leading in, the left path and the right, and that the right path leading out is the left path into one of the switches below. The left path of the top switch leads into the right path of the other, the switch below and to the left. If you drop a ball down the righthand path, it hits the trigger and flips that switch to the right. Then it continues down, hits the lefthand trigger below and flips that switch as well.

In the meantime, the ball on the platform is set in motion (when the switch is flipped) and then hits the trigger. The top switch is reset to point to the left. The second ball then drops a level to the platform below, where it stops. The playing field is composed of five levels, with four switches in the first level and eight in the bottom level. At the beginning of the game, there are no balls on the field—all platforms are empty—and the position of each switch is chosen randomly.

Moving Down The Path

Players alternate dropping balls into one of eight entry points. These balls (and others) may or may not make it all the way through the switchbox, to one of the 16 exit paths. Balls fall straight down (with one exception), so a ball's movement is always predictable. When it hits an empty switch, one of two things can happen. If it lands on the empty platform, it stops dead in its tracks. But if it lands on a trigger, it falls through to the next level below.

Moving balls always make it through loaded switches. Triggers allow balls to continue, and move the switch to the other position. If it's loaded, the dead ball on the platform is put into motion and it hits the trigger that just moved over.

This makes the switch go back to its original position, but with an empty platform. So when a ball hits the trigger of a loaded switch, its motion continues unabated. The switch moves, the ball on the platform begins to fall and it hits the newly placed trigger. The newly emptied switch moves back again, and the two balls drop to the next level.

There's one more possibility: a ball dropping onto a platform that already holds a ball. A platform can't hold any more than one ball, so when this happens one of the balls slides over to the trigger. So the ball does not move straight down—it slides over to the next pathway. This is the exception to the rule that balls drop in a straight line. Of course, when the ball hits the trigger, the switch changes position, causing the other ball to drop and hit the trigger.

The Chain Reaction

At the game's start, all platforms are empty, so four of eight entry paths are blocked. Remember that your turn ends when a ball hits an empty platform and stops. As the switches fill up, the chances increase that a ball will descend through several levels. The goal is to score points by getting balls to pass all the way through the maze of the switchbox. The best way to collect a lot of points is to cause a chain reaction.

A ball that hits a loaded switch from either side continues on its way. And the previously inert ball on the platform starts moving. One enters, two exit. If both of those balls encounter full platforms, four drop from the switches. The pathways are staggered, so the effects can spread outward, with more and more balls cascading toward the bottom.

Rather than taking an easy point or two, it's often worthwhile to build up layers of loaded switches. Watch out for leaving yourself vulnerable, though. Because players take turns, you'll want to leave positions where your opponent's move gives you a chance to create a chain reaction. The best strategy is to play defensively. Look ahead a move or two, and watch for an opening that allows you to score several points at once.

Four Quarters

A game of Switchbox always lasts four rounds. In the first (equality), each exit counts for two points. Your goal is to score ten points. The second quarter has more points available, as well as a higher goal. If you look at the exits, you'll see that the further away from the middle, the higher the point value. The numbers increase in a "Fibonacci" sequence: 1, 2, 3, 5, 8, and so on. Each number is the sum of the previous two (1+2 is 3, 2+3 is 5, 3+5 is 8, etc.). The target score in round two is 40.

In round three the numbers are a bit lower. They increase arithmetically (1, 2, 3, 4, up to 8 in the corners). A goal of 20 points brings you to round four, where you can score big. Here the numbers are squares: 1, 4, 9, 16, 25, all the way to 64 at the edges. In rounds two through four, it's sometimes prudent to leave a middle path open for your opponent to score a few points, in order to gather a high score on the big numbers to the left and right.

Each round lasts until one player has reached the goal. At that point the other player has one last turn before the round ends. It's possible to win the round on this last-chance play; watch out for barely topping the goal and leaving a chain reaction open for the other player. An arrow points to the scoreboard of the player whose turn it is. On the other side of the screen, you'll see a number where the arrow should be. That's the goal for the current round (the Amiga version displays the goal on both sides of the screen, below the scoreboards).

Bonus points are awarded at the conclusion of each round. Four numbers appear below the scorecards. The first is simply the total so far. The second is the total plus a bonus of the goal for the round if the player's points are equal to or greater than the goal. For example if the goal is 20 and you get 18, there's no bonus. If you score 22, the bonus is the goal for that round (20) and you'd have 42 points. The third number under the scoreboard is the difference between scores for the rounds. If you win by two points, two is added to your score (and two is subtracted from the other player). The final number is the grand total of the first three scores and bonuses. Rounds one and three are fairly low-scoring with low goals. You may want to seed the field with extra balls during these quarters, so you can collect more points in the second and fourth quarters.

Variations

Although the goal of the game is to score the most points, there's no reason you couldn't agree to play for low score. In a "lowball" game, you would try to avoid scoring points. You wouldn't necessarily play backwards, you would have to adjust the strategy of where to place the balls. Fill up the board as much as possible and leave your opponent in a situation where he or she is forced to score points.

The DATA statements at the beginning of the program (the *Setup:* routine in the Amiga version) determine the goal for each round and the point values for the exit paths. You can prolong the game by doubling the goals; this also dilutes the value of a big score at the beginning of a round, preventing one player from winning on the first or second turn. An interesting variation is to assign negative values to some slots. If some paths score negative points, you are forced to think harder about where the balls will drop.

In addition to the numbered keys (1–8), the plus (+) and minus (−) keys are active. Pressing plus drops a ball at random down one of the eight entry paths. Pressing minus allows you to pass your turn to your opponent.

Once you've mastered the regular game, you can add some new rules. Each player gets three passes per half, similar to the three timeouts in a football game. If you don't like the looks of the board, press the minus key to use one of your passes. After one player has skipped a turn, the other player must play (this prevents the possibility of six passes in a row). It's also a good idea to make a rule that a player can't pass on two consecutive turns. You can also give each player two random moves to be played for the opponent. In other words, after making a move, you

could inform your opponent that you're going to give him one of your random moves and you would press the plus key.

Here's one more change you could make: Instead of alternating turns, allow a player to continue after scoring. When a player drops a ball and scores some points, the other player would have to pass (by pressing the minus key). If the first player scores again, the opponent passes again, and so on until no more points are scored.

Playing Solitaire

To drop a ball, press a numbered key (1–8). If you're using a 128, ST or Amiga, the numeric keypad is convenient for choosing a move. By using the pass and random turn options, you can play against the computer. Here are the rules for solitaire play:

1. The computer always scores first. At the beginning of every round, the computer plays randomly until at least one point is acquired. Press the plus key for the computer's turn. You must continue passing (skip your turn with the minus key) until the computer puts points on the board.

2. After the first score by the computer, you can begin to play. When the computer has a turn, press the plus key for a random move.

3. Whenever you make points, you must pass again until the computer scores. When the computer gets more points, you can begin to play again. This rule means you should hold back on the easy scores of a few points; wait until there's an avalanche available.

4. If you're the first to reach the goal, the computer gets a last chance. Don't make this move randomly; figure out the best opportunity for scoring and play that move for the last-chance turn.

In the interest of keeping these programs to a manageable length, no attempt has been made to provide an "intelligent" computer opponent. Once you become familiar with the game, you might find it an interesting project to try adding some routines that give the computer a rational basis for picking one move over another. ©

Prisonball

John Scarborough
Version by Tim Victor

Nearly everyone has played Pong *or* Breakout, *two computer-game classics. "Prisonball" creates an intense, two-player action game by drawing features from both of those games. A set of paddles is required. Disk File: PRISON.BALL.*

"Prisonball" is a two-player action game that combines the best elements from two classic computer games, *Pong* and *Breakout*. The object of the game is simple—knock out as many bricks as you can in the allotted time.

Break To The Center

The game begins by displaying five colored walls running vertically down the center of the screen. Each player controls two paddles located at the left and right sides of the screen. Three balls appear at a random location and start bouncing around the screen. When a ball is on your side of the screen, move one of your paddles into its path to deflect the ball toward the walls. You can only hit a ball when it's moving toward your paddles (away from the interior walls). Balls travelling from the opposite direction go right through your paddles. If you happen to miss a ball, it wraps around the screen and appears on the other side, giving your opponent a chance to score.

At the beginning of the game, all three balls are a neutral color. Each time you hit a ball, it changes to the color of your paddle. You score whenever a ball of your color hits one of the five interior walls. The score depends on which wall you hit. The center wall is the hardest to reach, so it yields the most points. The two intermediate walls are worth less than the center wall. The outermost walls are easiest to hit and score the fewest points.

The top of the screen displays each player's score and a countdown timer. When the timer runs to zero, the game ends and the player with the most points wins.

Every time a ball hits one of the walls, a brick is knocked out of the wall at the point of impact. By aiming your shots carefully, you can bore a path through a wall and move a ball into the interior space between two walls. When this happens, the ball bounces wildly back and forth between the walls, scoring many points in a short time.

An additional bit of strategy has to do with the redrawing of walls. Whenever a wall has been destroyed, it is immediately redrawn. Some of the highest scores result when you trap one or more balls behind a wall when it is redrawn. Since the wall is new, the trapped balls may hit it many times before they break back out to the exterior.

Start the game with a **BRUN PRISON.BALL** command. Prisonball is played with paddles and each game lasts three minutes. If the action becomes too hectic, press any key to pause the game. If you wish to quit the game and exit to BASIC, press CTRL-C.

Prisonball uses a special technique to put a text window at the top of the lo-res screen. You may need to remove the parallel printer interface from your computer in order to make this work. ©

Backgammon

Jeffrey Vavasour and Geoff Rideout
Apple version by Tim Victor, Editorial Programmer

The classic game of backgammon combines equal parts of luck and strategy. This version observes all the regulations of standard backgammon. For the Apple II+, IIe, and IIc. Works with both DOS 3.3 and ProDOS. Disk File: BACKGAMMON.

The computer's your opponent in this challenging game of backgammon. Using a decision-making algorithm, the computer plots its best moves while you develop your own winning strategy. The only thing missing is the doubling cube, sometimes used in gambling.

Type in and save "Backgammon" to disk. If you're playing on an Apple IIe or IIc, be sure that the Caps Lock key is pressed down.

If you're already familiar with the rules of backgammon, you can skip over this next section.

The Rules Of Backgammon

The best way to learn backgammon is to have someone who plays teach you. If that's not possible, you can probably find books about backgammon at a local library. Here's a brief introduction to the basic rules:

- The playing board contains 24 *points*, divided into four groups of six, with a bar running down the middle. Players take turns moving their pieces around the board, and on the points. One player moves clockwise, the other counterclockwise. Your goal is to move all your pieces into the *inner table* and then start moving the pieces off the board. The inner tables of the two players are directly opposite each other. On the screen, you move clockwise from X to A toward your inner table in the upper-left corner (the positions labeled A–F). The computer moves counterclockwise toward positions S–X.
- When it's your turn, you (the computer, really) roll two dice. Say you get a 3 and a 5. You can move one piece three points and another five, or you can move a single piece five and then three, for a total of eight. Any piece can be moved from one position to another as long as the destination is not *blocked*. If your opponent has two or more pieces on a point, it's considered blocked and you can't move there.
- If there's a single piece there, you can *blot* it—your piece takes the position and your opponent's blotted piece is moved to the bar in the middle. When one or more of your pieces is on the bar, you can't make any moves on the main field until the piece moves off the bar back onto the board. You have to move it to one of the points in your opponent's inner table, based on the roll of the dice. In this game, for example, if you have a piece on the bar and roll a 1 and a 5, you could move off the bar to position X (to use the 1) or position T (5). Then, you'd use the other number to move a single piece.
- Rolling doubles gives you double the moves. So if you roll two 3s you get four moves of three points instead of two moves of three points.
- Once you've moved all your pieces to the inner table—the six points labeled A–F—you can start to *bear off*. Rolling a 2 and a 3 would allow you to remove one piece from B and one from C. The first player to remove all pieces from the board wins the game.

A lot of backgammon strategy is building safe positions, points containing two or more pieces. In general, you should not leave a single piece on a point where it can be blotted by the other player, although sometimes it's unavoidable. It also helps to have an understanding of the odds for rolling certain combinations with the dice.

Basic Game Play

Throughout the game, a graphic representation of the backgammon board is displayed on the upper portion of the screen. Board positions consist of the letters A through X (representing the board locations, starting in the upper-left corner and running counterclockwise) and BAR. Board positions are printed next to their appropriate locations.

When you run the program, you're asked *WILL I GO FIRST?* If you want the computer to move first, respond with *Y*. Otherwise, press *N*. The computer draws the board and places the game pieces in the starting position (you start with five pieces in your inner table, two at the far end in the computer's inner table, and eight in between). The computer's pieces are not highlighted, and move counterclockwise from A to X. The highlighted pieces are yours. They move clockwise from X to A.

When it's your turn, the message *PLAYER ROLL* is displayed below the game board, followed by the result of the dice roll, based on a random throw. If you roll doubles, the number is printed four times to indicate that you have four moves. Below this you're presented with a *MOVE FROM?* prompt. Respond with the location you're moving from by typing the appropriate letter A through X. If the location you entered doesn't contain any highlighted pieces or is not within the range of the board, *ILLEGAL MOVE* is displayed below the prompt and the cursor returns to the beginning of your input. If you have any pieces on the bar, *MOVE FROM BAR* appears and the computer automatically skips to the *TO?* prompt.

After you've responded to the FROM? prompt, *TO?* appears. Answer this with a letter from A to X representing the destination on the

board, or *OFF* indicating you wish to take the piece off the board. If your move is not permissible, *ILLEGAL MOVE* displays and the cursor returns to the FROM? prompt.

If you can't move, typing *NO* at either the FROM? or TO? prompt skips the rest of your turn and starts the computer's.

To move off the board, *all* your remaining pieces *must* be on positions A through F, in other words, in your inner table.

When it's the computer's turn to move, the input area below the board is cleared, and the computer's rolls are displayed in the same form as the player's.

While the computer is evaluating each of its moves, *THINKING...* is displayed. The message *MOVE FROM xxx TO yyy* follows after a few seconds, indicating the computer's decision.

Once the computer has exhausted all of its moves, after it has used all of the dice or when it can't move due to some block (in which case *CAN NOT MOVE* is displayed. This gives you a chance to look at the move made by the computer. Press any key to proceed to your turn.

When the game is over, the computer displays the winner and offers you a chance to play again.

How It All Works

The computer's strategy uses a process of evaluation and elimination. The program starts at the bar and scans the board through location *X* using each of the die rolls. It then records the highest evaluation number (described below), highest evaluated move, and roll. If there's no evaluation number (it's 0) or there are pieces on the bar and the evaluation is less than 128, the computer decides it can't move. You're told this and the computer's turn ends. Otherwise this process is repeated until all rolls are used.

A particular move is evaluated by adding up various values reflecting the advantage of such a move. The computer evaluates a move off the bar as 128 (being of most importance) and off the board as 64. Protection of two exposed pieces by moving one atop another has a value of 32. If the move doesn't expose any other pieces, either at the origin of the move or the destination, a value of 16 is added. If the opponent can be placed on the bar, the evaluation is 8. Movement on the outer table is counted as 4. Also, moves where another piece covers that being moved are given a value of 2. Legal moves are assigned a value of 1, a method to determine whether any legal moves were found at the end of the evaluation process.

Any of these values may be combined to form a general evaluation of the move. For instance, a move off the bar which places the other player *on* the bar is given a value of 136 (128 and 8). The table included in the sidebar contains a list of move values, a brief description, and the program line which contains each.

By changing these values, you can change the computer's strategy, increasing or decreasing the difficulty of play or just adding some variety to the computer's logic. If you want to change a value, simply replace the number following the statement $E=E+x$ in the appropriate line. When doing this, it's best to leave moves off the bar unchanged because a value less than 128 will prevent the computer from being able to move off the bar.

If you're a backgammon expert, you may find that you can beat the computer most of the time. In playing, we tend to take calculated risks which may eventually pay off. We can also look one or two moves ahead, something the computer can't do. The computer's process of evaluating every possible move makes it avoid risks—if at all possible—which results in a fairly conservative game.

Variable Descriptions

Variable	Description
A$	Used when waiting for a key at the end of the computer's turn.
B(x)	Board contents. B(0) is computer bar, B(1–24) represent board locations A through X, and B(25) is the number of pieces off the board. A positive number indicates computer pieces; a negative number indicates player.
E	Evaluation of current move being processed by the computer.
F$	Input from player's FROM? prompt.
H	Highest evaluated move by computer.
HP	Computer's highest evaluated position (0–24) or player's FROM position (1–25).
HR	Computer's highest evaluated roll or how far player wishes to move.
LN	Die number (0–3) of lowest computer roll.
LR	Value of lowest computer roll, same as R(LN).
MF$	Indicates current turn. If string starts with *N*, it is the player's turn; otherwise, it is the computer's turn.
OB	Nonzero indicates computer may move off board. Also contains the die number the player is using (HR usually equals R(OB)).
PB	Indicates the number of pieces the player has on bar (−2 means two pieces on bar).
PO	Indicates the number of pieces which the player has moved off bar.
R(x)	Contains the remaining rolls. Used die are indicated by a 0; *x* must be from 0 to 3.
FNR(x)	Returns a random number between 1 and 6.
T$	Input of player's TO? prompt.
X,Y,Z	Miscellaneous counters and pointers.
X$,Z$	Computer's FROM and TO respectively.

Computer Evaluation Ranking

Rank	Description	Line Number
128	Move computer piece off bar.	350
64	Move computer piece off board.	360
32	Move an exposed piece to a position also containing an exposed computer piece.	370
16	A move which does not expose any pieces.	380
8	A move which places the player on bar.	390
4	A move that is not in the computer's inner table.	400
2	A move in which the piece will be covered by another piece in the same turn.	450
1	A legal move.	420

Euchre

David Shimoda

Here's a finely detailed implementation of the popular card game of Euchre. Disk Files: EUCHRE and EUCHRE.BIN

"Euchre" is a four-handed translation of the popular card game of the same name. In this version, you play with a computer partner against two computer opponents. The computer will deal the cards, keep score, and play your partner's as well as your opponents' hands. Even better, it never gets bored or commits blunders such as trumping your ace. Nearly all the subtleties of the original card game are reproduced faithfully, including lone hands, short suits, and more. You can even choose different personalities for your partner and opponents.

Computer Personalities

The game begins by asking you to choose personalities for your partner and your opponents. Move the reverse-video cursor to your choices, and make selections by pressing the joystick button or the Return key.

The normal personality plays a more cautious game, while the aggressive personality tends to take more risks. Both opponents must have the same personality, but the partner's personality is chosen separately. This makes the game much more varied than if the computer players always stick to the same, predictable strategy. One of the more difficult combinations is to choose a normal partner and aggressive opponents. Of course, your own style of play will have an impact on which combination you prefer.

Dealing And Trump

This Euchre variation uses only 24 cards from the standard 52-card deck. Each suit includes only the 9, 10, jack, queen, king, and ace. (The ace is high.) Before actual play begins, the first dealer must be selected. This is done by dealing out cards until a black jack is thrown. The first person who receives a black jack becomes the first dealer. After each hand, the position of dealer passes to the next player in clockwise order.

The dealer deals out 5 cards to each player and then places 1 card, face up, on the center of the table. The program automatically deals the cards, as it handles many other details in this game. As a consequence of this scheme, only 21 of the 24 cards are in play for any given hand. (Three cards are always left unplayed.)

The next step is to choose *trump*; the trump suit is the most powerful of the four suits for the current hand. Trump is determined by moving around the table in clockwise order, giving each player an opportunity to choose whether the dealer should pick up the center card. Each player can either pass or *order up*—order the dealer to pick up the center card. When the dealer is forced to take the center card, that card's suit becomes trump, and the dealer discards one card. The computer players, of course, decide for themselves whether to pass or order up in this phase of the game.

If no player chooses to order up in the first circuit of the table, each player then has a chance to pick any other suit as trump. If no player chooses trump on the second circuit, the hand is thrown out completely, and another is dealt.

Lone Hands

On certain occasions, a player may choose to exclude his partner from play, a tactic which is known as playing *lonehand*. The player who chooses trump must choose at the same time whether or not to play lonehand. If a player orders up a card into his partner's hand, the player who ordered up must play lonehand. (If your partner is the dealer and you order up, you must play lonehand).

For instance, you might want to play lonehand in a case where you hold most of the high cards in a suit, and your partner is the dealer, and the center card is a high card of your strong suit. By excluding your partner and playing lonehand, you are in a very strong position to take most or all of the tricks.

Tricks And Hands

A hand consists of five *tricks*. A trick consists of all players laying down one card. The player to the left of the dealer throws down the first card in the first trick. Subse-

quent tricks are begun by the winner of the previous trick. Suit must be followed within a trick. That is, you must throw a card of the suit which was led, as long as you have any card of that suit.

If no trump cards are thrown in a trick, the trick is won by the player who laid the highest card of the leading suit. If trump is thrown, then the highest trump card takes the trick.

For all suits except the trump suit, the rank of the cards follows the usual order. (The 9 is low, and the ace is high.) For the trump suit, however, the jack is the highest-ranking card. The jack of the same color, but different suit, is considered part of the trump suit—and it is the second highest ranking card. For example, if the trump suit is chosen as clubs, it follows this ranking:

jack of clubs
jack of spades
ace of clubs
king of clubs
queen of clubs
10 of clubs
9 of clubs

A hand is won by the side which wins a majority of tricks (three or more). If you or your partner orders up a card, your side must take the majority of tricks in that hand or else be *euchred*, meaning that the opposite side gets two extra points.

Scoring

A game of Euchre ends when one side accumulates ten or more points. You score one point for winning a hand, two points for winning all the tricks in a hand, and four points for winning all the tricks lonehand.

Euchre runs on any computer in the Apple II series, under either ProDOS or DOS 3.3. The program consists of two files. The main program (EUCHRE) is written in Applesoft BASIC. It automatically loads a second file named EUCHRE .BIN which is used to create graphics. The game is played with keyboard controls. Use the arrow keys and Return key to move the cursor and make selections. ©

Solitaire

Ben Elizer

Looking for something different to do with your computer? Like the conventional game of solitaire, this computerized version requires you to think ahead at all times. Disk File: SOLITAIRE, SOLITAIRE.BIN.

"Solitaire" is an electronic version of the familiar card game. Like the original, this game challenges you to put a deck of cards in order using the fewest possible moves.

Unshuffling The Deck

As you probably know, Solitaire has a very simple object. After shuffling a deck of playing cards, you must put them back in order, following a few simple rules. Though there are several different variations of the conventional game, here are the rules for this version:

When you run the program, the computer deals out four rows of 13 cards, then removes the aces, leaving four empty spaces. Your goal is to rearrange the cards into four rows of the same suit, putting the cards in each row in ascending order from the lowest (2) to the highest (king), without leaving any empty spaces between cards. That sounds simple enough. But since you must move a card into one of the four empty spaces, your choices for any given move are limited.

Your position on the screen is shown by a blinking cursor. Press the M key to move from the current position to another empty space. When you press P, the computer moves a card into the current space: Which card it puts there depends on which card is immediately to the left of the space. Whenever possible, the computer uses the next card in suit. For example, if the card to the left of your current position is the 2 of hearts, pressing P puts the 3 of hearts in the current space and puts a space where the 3 of hearts was before. If you press P on a space to the right of the queen of diamonds, the king of diamonds moves from its current position to that space, and so on. Each time you press P, one space is filled and another is emptied.

In this way you can gradually move cards into the right order. When you press P on a space at the beginning of a row, the computer asks which suit to play (hearts, clubs, spades, or diamonds). This determines the suit for that row. While it's possible to win on only one deal, most games require two or more deals. When no moves are possible (every empty space is followed by a king or another space), the computer automatically shuffles the remaining cards and deals them out again. Of course, it does not disturb cards that are already in correct order. You'll find that it takes considerable foresight to win consistently in only two or three deals. Completely random play results in an average of nine or ten deals.

The program works on any Apple II-series computer, but the graphics look much better on a color monitor. ©

Biker Dave

David Schwener
Version by Tim Victor, Editorial Programmer

This machine language game is guaranteed to bring out the daredevil in you. Settle in the seat, twist the throttle, and get ready for the motorcycle ride of your life. Works with all Apple II computers in either DOS 3.3 or ProDOS. Disk File: DAVE.

As the ramp nears, you focus your mind, tighten your grip on the handlebars, and accelerate the motorcycle for the final approach. The deep, throaty cry of your machine's powerful engine drowns the spectators' cheers, and the onrushing wind pushes against your body like a gigantic hand. If your speed and timing aren't exactly right, you may overshoot the ramp and lose control, or fall short into the line of cars.

Will you earn fame by surviving the jump or tumble into anonymity with a cartwheeling crash? As your speed mounts and the sidelines fade into a blur, there's no more time to wonder and no chance to turn back. Only the utmost in coordination and skill will bring you safely to earth on the other side.

"Biker Dave" is a realistic computer game that simulates the thrills and challenge of motorcycle acrobatics.

Over The Ramp

To play Biker Dave, type **BRUN DAVE** and press Return.

You'll first see the scoring screen, which tells you the currect score (if you've just started the game, it will be 000), the cars you've jumped (00), and the bikes you have left (5).

Press any key, and the game screen appears. Your rider and motorcycle are in the upper left corner of the screen. The rest of the screen contains the racetrack, with a number of tunnels (the blue blocks) along the way.

The most formidable obstacle consists of several autos flanked by launching and landing ramps near the bottom right. Press the space bar to accelerate the bike. Your goal is to ride down the track, through the tunnels, and toward the final obstacle, gaining just enough speed to jump over the cars without crashing.

There's also a large, fiery hoop midway through the course. The hoop has a launching ramp, but no landing ramp. Each time a car is added to the final obstacle, the hoop's launching ramp moves farther away, too.

That may sound easy, but it's not as simple as you might think. For one thing, your bike is a specially built stunt machine with no brakes. There's no way to slow down. And if you accelerate too fast, the bike pulls up into a wheelie. That's not bad in itself, but if you accelerate too hard from a wheelie, the bike tips backwards and crashes. And you go skidding across the pavement.

As you approach the launching ramp, you need to go just fast enough to clear the parked cars, but not so fast that you lose control and miss the landing ramp on the other side. A successful jump requires precise timing and sure control of the throttle. The score you earn depends on the number of cars jumped and the number of attempts you made at that level.

Each time you jump over the cars, the racetrack crew moves the launching ramp. When it has been shifted far enough, the crew also adds another car to the lineup. Unfortunately, the crew is somewhat unreliable and has been known to change the launching ramp's angle slightly when moving it. Thus, even though you may have jumped three cars at a speed of 100 miles per hour, there's no guarantee that the same speed will work every time.

When you crash, press the space bar and the scoring screen appears, just to remind you of the score, cars jumped, and bikes left.

Reflection

Sean Puckett
Version by Chris Poer

"Reflection" is a fast-paced computer version of reversi. You can play it as a strategy game with two people, challenge the computer, even set up your own game board. For all Apple II–series computers in DOS 3.3 or ProDOS. Disk Files: REFLECTION, REFLECTION .LDR.

Through the ages, people have devised many pastimes to exercise their minds. The best-known match of wits is chess, with backgammon and checkers running close behind. Another board game, reversi, though perhaps not as popular now as when it was introduced in the late 1800s, does manage to combine the logic of chess, checkers, and backgammon with the action and excitement of a good football game.

The trouble is, some players can become so excited that they tend to get carried away and attempt a forward pass with the board, or they fumble and scatter the chips everywhere (a method often employed by sore losers). A computer version of reversi is ideal. The computer can act as referee, permitting only legal moves, or it can be a ruthless opponent.

"Reflection" gives you the option of playing either way—against another person or against the computer. The rules are quite simple. Players take turns placing chips on the board, one piece per turn. To capture your opponent's pieces, you sandwich a row of them between one of your existing pieces and the one you're laying down. You can capture one or several pieces this way. The row can be vertical, horizontal, or diagonal. Once a piece is captured, it turns to your color and becomes (at least temporarily) one of your pieces.

The best move is either the one that captures the most pieces or the one that leaves your own pieces less vulnerable—depending on the stage of the game. Sometimes you can place a single piece to capture more than one row of chips. Each player must capture at least one enemy piece per turn, or the turn is forfeited. Corner and edge positions are valuable, since these are difficult if not impossible positions to surround. When all of one player's pieces have been captured, or when neither player can make a legal move, the chips are tallied and victory is awarded.

Because capturing an enemy piece converts it to your color, the game can reverse directions very quickly. Even if you think you're winning, you can suddenly find yourself far behind, with most of your chips flipped to your opponent's color. Reversi's sudden reversals keep players on their toes. You must plan several moves ahead, since a seemingly triumphant capture can just set you up for devastation.

Playing Reflection

Before you begin play, you have several options:

W(hite) moves first
B(lack) moves first

N(ormal) game board
D(ifferent) game board

O(ne) player
T(wo) players

What level (**1**–**2**)

Computer plays **B**(lack)
Computer plays **W**(hite)

Press the appropriate keys as the options appear. You can choose which player goes first, what board you'll use, whether you're playing against the computer (one player) or against another person (two players), the level of play (only if you're playing against the computer), and what color the computer takes (again, only against the computer).

If you're playing on the normal game board, the grid then appears, with two black and two white chips in a diamond-shaped pattern in the middle of the screen. A cursor, which repeatedly enlarges and contracts, is also visible.

If you're playing Reflection on an Apple IIe or IIc, make sure the Caps Lock key is pressed down. Use the I, K, M, and J keys to move the cursor up, right, down, and left, respectively. Press the space bar to put down your piece. You can put down only one piece per move, and only on empty squares. If you place your chip so that it doesn't capture any enemy pieces, the program removes the piece and you forfeit your turn. You must purposely forfeit in this way if you can't make a legal move. If neither player can make a move, press E to end the game.

Reversi Options

When playing against the computer, there are two levels of computer intelligence. Level 2 plays better, but it takes longer for the computer to make up its mind.

You can also set up the board yourself, placing as many pieces as you want. This is a good way to restart an unfinished game (assuming you noted the pieces' positions) or to study a particular reversi problem. After choosing D from the option menu, an empty grid will appear. Press the W key to set down a white chip, B for a black chip, and space to skip a square. You continue left to right, top to bottom, until you reach the lower-right corner.

To end the game, simply press the E key.

Two Programs

Before you load Reflection, you need to enter two POKE statements in direct mode:

```
POKE 104, 64: POKE 16384,0
```

To make it easier, you can run "Reflection Loader." It does the POKEs for you, then loads Reflection from disk. Just make sure that Reflection is on the same disk as the Loader, and that Reflection is named exactly that. ©

Wari

Don Donati
Version by Patrick Parrish, Programming Supervisor

"Wari," based on an ancient strategy game, lets you match your wits against the computer. Disk File: WARI.

"Wari" is a strategy game which has been played for centuries in Africa and the Middle East. The object of the game is to capture as many of your opponent's pieces as you can, while trying to prevent the capture of your own pieces.

Electronic Beans

Wari is played on a board which has 12 compartments arranged in two rows of 6 (the arrangement is similar to that of an egg carton). In the original versions of this game, the compartments were actual depressions in a board or simply holes scooped into the ground, and the game was played by moving counters (beans, pebbles, or other small objects) among the various compartments. In the computerized version of Wari, the compartments are rectangles drawn on the screen, and the counters are represented by numbers. If the number 4 appears in a compartment, that compartment holds four counters, and so on.

Six of the compartments are yours and the other six belong to your opponent, which is always the computer. When the game begins, four counters are placed in each of the 12 compartments, for a total of 48 counters. Each player, then, starts the game with 24 counters. Once play begins, however, counters are considered yours when they rest in one of the six compartments on your side of the board. The program then asks whether you or the computer should make the first move.

Counterclockwise Movement

A move consists of taking all the counters from one compartment on your side of the board and sowing, or distributing, one counter into each of the adjacent compartments in a counterclockwise direction. In the original game, this was done by picking up the counters and sowing them by hand. In this version of Wari, you simply indicate which compartment you wish to sow by pressing the letter key printed by that compartment. The computer automatically sows that compartment's counters for you.

The figures illustrate the effect of making a move at the beginning of a game in which you have chosen to move first. Figure 1 shows the board before a move is made. Each compartment contains four counters, represented by the number 4.

Figure 2 shows the board after you press the G key. The four counters from compartment G are sown in compartments H, I, J, and K. Each of those compartments now contains five counters; the original compartment (G) is now empty.

Depending on which compartment you sow, the sowing can wrap around from one side of the board to the other. For instance, say that your next move sows the counters from compartment H: Compartments I, J, and K now contain 6 counters; compartment L contains 5 counters, and the fifth counter from compartment H is sown in compartment A. Should you sow 12 or more counters, you return to the compartment where you started: In that case the original compartment is skipped and sowing resumes in the next one.

Figure 1: Before The First Move

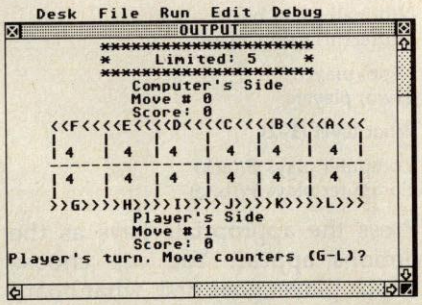
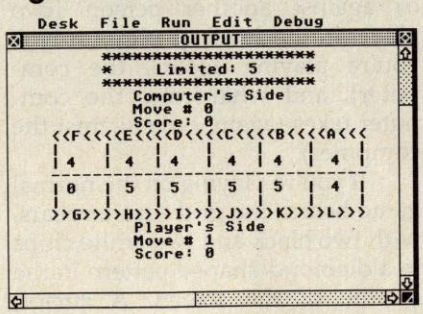


Figure 2: After The First Move



Captures

You score points in Wari by capturing counters. A capture occurs when you sow your last counter in an opponent's compartment which previously contained either one or two counters. The counters from that compartment are then removed from play. Each captured counter is equal to one point. If the previous compartment in line also contains two or three counters at the end of the move, its counters are captured, as well. This process continues until no more counters can be captured in that turn (in some cases, it's possible capture all of your opponent's counters in a single move).

The game can end in several different ways. Play must end whenever the board is empty (all counters have been captured) or whenever one player has no more counters to move. The game also ends when one player has captured more than half of the counters (if you have more than 24 counters, it's numerically impossible for your opponent to win the game). Wari also permits a stalemate, where each player chases the other around the board fruitlessly; a stalemate game should be ended by mutual agreement.

At the end of the game, your score is increased by the number of counters remaining on your side of the board. The computer automatically totals the score and announces the winner. You may end the game at any time by pressing the *Q* key.

Game Limits

At the beginning of each game, the computer also asks whether you wish to play a limited or unlimited game. A limited game consists of a set number of moves for each player; an unlimited game lasts until the game ends in one of the ways described above. Press L for a limited game or U for an unlimited game.

Limits of 25–35 moves make for interesting games. In a very short limited game (say, five moves) it's usually more important to protect counters on your side of the board than to capture the opponent's counters (when the game ends, you are awarded all the counters on your side). ©

Climber 5

James Rogers

As ballboy for a baseball team, it's your job to retrieve a home-run ball from the top of a nearby building. Time is of the essence, but you still need to avoid moving obstacles on your way to the top. Disk File: CLIMBER5.

"Play ball," the umpire cries, and the game begins. As ballboy for the home team, your job is easy: Retrieve stray balls and return them to the umpire. The number 5 on the back of your uniform shows that you're part of the team, and your job is more important than some might think, since it so happens that this game is being played with only one baseball. If a ball flies into the stands or out of the park, the game grinds to a halt while a stadium full of fans and players waits for you to bring it back.

You're used to the pressure, but that doesn't make the waiting any easier. The first inning passes, followed by the second and third, without any problem. Then, at the top of the fourth inning, the first batter swats a towering homer over the left-field wall. Up, up it goes, so high that you grab your field glasses to track it. Yes, it's outside the stadium—looks like a long sprint to get it back. Oh, no! Did that really happen? The ball lodges at the top of a neighboring building which is still under construction. As you rush to retrieve the prize, you'll have to duck and dodge to avoid obstacles on the construction site.

The Climber 5 screen consists of several horizontal levels. When the game begins, you are at the bottom right corner of the screen, and the ball is at the upper left. Your job is to climb and run to where the ball lies, avoiding all the moving obstacles along the way. Since some of the moving objects are speedy indeed, that's more difficult than you might imagine at first.

The game is played with keyboard controls. Press the I, J, K, or L keys to move up, left, down, or right, respectively. In this version, you can avoid objects by leaping straight up or by somersaulting to the left or right. Press the U or O key to somersault left or right, respectively. Press the space bar to jump straight up.

Each level is connected to the next by one or more ladders. The controls allow you to move up or down a ladder; of course, you must be aligned with the ladder in order to ascend or descend on it.

You have a total of five players when the game begins. Whenever you hit a moving object, you lose one player. The game ends when you have lost all your players. If you reach the ball without being hit, the program displays a congratulatory message and lets you try the next skill level, where everything becomes more difficult.

To start the game, use the command of the form **BRUN CLUMBER5.** ©

High Rise

Charles McGuyer
Version by Tim Victor, Editorial Programmer

You're a construction worker, trapped in a partially completed high-rise building after dark. Can you make it safely to the ground floor without being zapped by the patrol robot? This unique game, written entirely in machine language, is one of the best Apple arcade-style games we've ever published. Works with DOS 3.3 or ProDOS. Disk File: HIGHRISE.

The time is the not-too-distant future. The place is a downtown high-rise building under construction. You're just finishing the day's work when you realize it's already dark. Everyone else has gone home, leaving you alone in a shadowy, multi-story maze of naked girders and bare concrete. A chill creeps down your spine as you think about the recently installed antitheft robot. It patrols the structure from dusk to dawn, automatically disposing of any intruder.

Your only hope is to use the temporary elevators. They move randomly during the night hours, going up and down, stopping at some floors, skipping others. With skill and a little luck, you just might evade the dangers around you and make it safely to the ground floor—but it won't be a cakewalk.

Run Away

You can activate High Rise by typing **BRUN HIGHRISE.** The object of High Rise is to make your way to the ground floor via the elevators while evading the patrol robot. When the game begins, you'll see several floors of the building and a number of elevators moving up and down. Move your player with keyboard controls: Press the left-arrow key to move left, the right-arrow key to move right, and the space bar to stop.

You can jump on any elevator that comes to your floor (move into the elevator and it picks you up automatically), but there's no way to control its direction or how far it goes. These are just temporary elevators, used to transport materials and workers during daytime hours. The trick is to catch one that's moving in the direction you want and get off to catch another before it starts moving in the wrong direction.

When you reach the lowest floor shown, the screen scrolls up one floor, revealing the next lower level. Once you reach ground floor, the player sprints off the screen to safety and you can play another game.

The patrol robot always starts on an upper floor and moves systematically through the building, traveling up and down through special shafts that are closed to you. Designed to discourage theft and vandalism, the robot's technique is simple and effective: It pushes any intruders (including you) off the building. If it runs into an elevator and detects you inside, it sends a high-voltage charge through the elevator shell until you drop.

When boarding an elevator, make sure that your player is positioned exactly in the center of the elevator shaft (the double-dotted lines). If you're just a bit to one side, the elevator will not pick you up.

When the game begins, you have three players. Each time you get zapped or fall from the building, you lose a player. Play ends when all three have been lost. When the game starts, you're on the twelfth floor. Moving down a level earns you 100 points. If you reach the bottom safely, you'll have another chance to play, beginning at a higher floor. High Rise keeps track of the highest score attained in the current session, as well as your score in the current game. ©

Chain Reaction

Mark Tuttle

In this explosive strategy game, the contest is never finished until your last bomb has been thrown. Disk File: CHAIN.

"Chain Reaction" is a clever strategy game for one or two players. Whether you play against the computer or another human, the objective is the same: to eliminate all of your opponent's bomb-shaped pieces from the field of play. The game is played on a 5 × 6 grid of squares, and the players alternate turns, placing one bomb in a square on each turn.

The results of a move depend on how many bombs are already in the chosen square and adjacent squares. Whenever any square reaches "critical mass," it explodes and sends its bombs into neighboring squares. If those squares are already loaded to capacity, they explode too, creating a chain reaction that can engulf a large area of the board.

Bomb Begets Bomb

When you run Chain Reaction, it begins by asking whether you wish to play with one or two players. If you've never played before, you may want to play a game or two against the computer to learn what sort of strategies it favors. When you choose to play against the computer, the program also asks whether you'd like the computer to take the first turn.

The first part of most games involves placement of initial pieces, without many explosions. Use the I, J, K, and M keys to move the cursor up, left, right, and down, respectively. To place a bomb, press the space bar. As the board fills up, however, explosions occur with increasing frequency. Play continues until one player's pieces are completely eliminated from the board.

The position of a square in the grid determines how many bombs it requires to create an explosion. A corner square can hold a maximum of one bomb. When you place a second bomb in a corner square that already holds one, both bombs explode, sending a bomb of your color into two neighboring squares. After an explosion, the original square is emptied.

Other squares require more bombs to create an explosion. A border square that isn't on a corner can hold a maximum of two bombs. When you place a third bomb in a border square, its explosion sends three bombs into the squares that adjoin it. Squares in the center of the game board hold the most bombs and also create the most devastating explosions. When you place a fourth bomb in a central square, it sends four bombs into squares which adjoin that position.

When an explosion sends bombs into adjacent squares, any bombs in that square change color to match the color of the exploding bombs. Should one of the adjoining squares surpass its limit, that square, too, will explode, creating the potential for even more explosions. This process continues until no more explosions are possible.

Thus, the situation in Chain Reaction is often volatile. The lead frequently seesaws back and forth between players, as each creates increasingly more widespread chain reactions. Even if defeat seems almost certain, you can often regain the lead with clever play. When a game ends, the program announces the winner and permits you to play a new game or quit.

Like other games of strategy and placement, Chain Reaction rewards the player who can think ahead. At first, you may be tempted to start making explosions as quickly as possible. But that's not always the best long-term tactic. By spreading bombs of your color throughout the board, you may be able to survive chain reactions that would otherwise wipe you out. ©

Webster Dines Out

Walter Bulawa
Version By Tim Victor, Editorial Programmer

Tired of blasting invaders from outer space? This whimsical game is set in a very different world—the miniature jungle in your own backyard. Disk File: WEBSTER.

Guide Webster, the hungry tree spider, in his endless search for a square meal. Roving back and forth across his tree limb, he watches for bugs to appear in the grass below. When the time is right, he drops down on a strand of silk for a light snack, then climbs back up his web to look for more.

Unfortunately, this backyard paradise isn't quite perfect. The more Webster eats, the faster the bugs move, making it harder to find the next meal. Even worse, he's not the only one with an appetite—there's a giant grasshopper sharing the same hunting ground, stealing bugs when he can and making things difficult for Webster.

Webster Dines Out will run on any Apple II series computer. To load and run the game, simply type BRUN"WEBSTER" and press RETURN.

Your goal is to score points as quickly as possible. Each bug is worth 25 points and you get 50 bonus points for snaring two bugs in a single drop. Use the left and right arrow keys to move on the branch, and press the space bar to drop to the ground. Avoid colliding with the giant grasshopper—when that happens, Webster loses a life (and is carried bodily off the screen). The grass in the lawn grows higher as the game progresses, making your job more difficult. You can drop into the grass to snare a hidden bug, but be sure to keep track of the giant hopper, who might be lurking there as well. ©

Laser Strike

Barbara Schulak
Version by Tim Midkiff, Editorial Programmer

In this strategy game for one or two players, your mission is to locate and vaporize your opponent's hidden space force before your force suffers the same fate. The program runs under either ProDOS or DOS 3.3. Disk File: LASER.STRIKE.

"Laser Strike" is a strategy game based on several popular board games (Battleship is probably the most famous). However, unlike the board games, the action in Laser Strike occurs in outer space. Two players secretly deploy their spaceships around the galaxy and then try to locate the opponent's ships by firing laser strikes on the two-dimensional galaxy grid. The first player to find and destroy all the opponent's ships is the winner.

Laser Strike begins by asking each player to enter his or her name. If you wish to play against the computer, press RETURN without typing anything at the first name prompt. If you press RETURN at both name prompts, the computer plays the entire game by itself.

The program then displays two grids, one for each player. In the first stage of the game, each player decides where to locate the ships in his or her grid. The deployment must be secret, so the second player needs to look away from the screen while the first player deploys ships, and vice versa. To deploy a ship, simply move the cursor to the desired location on the screen and press the space bar. After choosing the location, you must also decide whether to deploy the ship horizontally or vertically. You cannot place a ship so that it overlaps the border. If you attempt to place a ship illegally, the program warns you and gives you another chance.

After both players' ships have been placed, the contest begins. The players alternate firing laser shots on each other's grids. To fire a shot, move the cursor to the desired location; then press the fire button or the space bar. If you hit a ship, that square of the grid is marked in the color of the ship you hit. If you miss, the square is marked in a neutral color. The game continues until one player has found and eliminated all of the other player's ships. At that point you can exit the program or play a new game. ©

Mindbusters

Ned W. Schultz

Here's a graphics puzzle game that is both challenging and unusually fascinating. Disk File: MINDBUSTERS.

Are you ready to pit your brain against the computer's? "Mindbusters" presents you with three graphics puzzles that are guaranteed to keep your mind's microprocessors and memory chips whirring for hours.

You can choose to solve a mind bender, a mind bruiser, or a mind blower. Warm up with the mind bender—it's the easiest. When you're prepared to press your brain to its limits, you're ready for the mind blower.

Following your selection, the program constructs a puzzle and displays it at the upper-left corner of the screen. Your job is to match that puzzle in the workspace at the lower-right corner of the screen. What's more, you try to solve the puzzle in as little time as possible. A timer ticks away as you work. There's no limit to how much time you can take, but the timer lets you compare your progress to a previous performance, or against another player if you wish. Your fastest time during the current session will be displayed on the screen.

Each puzzle is composed of several horizontal rows of odd shapes. A tiny arrow to the right of the workspace points to the row you're currently working on. To work on different rows, you can move the arrow up and down with the I and M keys. To move the row of shapes next to the arrow left or right, press the J or K key. When you think you've matched a row to the puzzle pattern, start working on another row.

When you succeed in correctly matching all the rows, the program automatically signals that you've solved the puzzle. Then you can play again if you like.

Helpful Hints

Because Mindbusters can generate a tremendous number of different puzzles, there are very few tricks to mastering it. I suggest you work from top to bottom or vice versa. The best tip I can offer after hours of my own mindbusting is to concentrate, concentrate, concentrate. ©

The Witching Hour

Brian Flynn

This game of skill and foresight is ideal for a bleak, stormy October night. Disk File: WITCHING.HOUR.

When autumn winds send a shiver down your spine and the witching hour draws near, there's no better entertainment than a good computer game. "The Witching Hour" is an absorbing contest of strategy based on "Alquerque", a board game played in ancient Egypt and still popular in Spain today.

The Witching Hour pits broomstick-straddling witches against ethereal ghosts and is played on a board of 25 squares with 12 pieces to a side. After choosing sides, you attempt to take your opponent's players by jumping over them. You can move vertically, horizontally, or diagonally. However, certain diagonal moves are illegal (the lines between squares show where you can go) and only one square is vacant when the game begins.

Jumping an opposing player's piece removes that piece from the board. If no capture is possible, you may move any piece to an adjacent empty square. You may not pass up a capture—if it's possible to jump an opponent, you must always do so. The computer won't let you make illegal moves.

Play ends when all the pieces from one side have been removed from the board. You can play against a friend or measure your skills against the computer. Like other contests of strategy, The Witching Hour is simple to learn, but a challenge to master, and can be played at many different levels. Hint: It's sometimes smart to sacrifice a player to draw the opponent into a dangerous position.

The Witching Hour runs on any Apple II-series computer with DOS 3.3 or ProDOS. When the program starts, you must choose between a one- or two-player game. Then the game board is drawn and play begins. The flashing box shows which square you are on, and is moved with keyboard controls. Press the I key to go up, J to go left, K for down, and L for right. Press RETURN when the box is on the piece you want to move, then move the box to the desired square and press RETURN again. ©

Heat Seeker

Tim Victor, Editorial Programmer
(Original game concept and VIC-20 version by Jeff Wolverton)

Your jet climbs upward to avoid the missile, then dives for the ground. But it's still on your tail. You can't shake a heat seeker. A fast-action, machine language game for Apple II series computers. Joystick or keyboard control. Disk File: SEEKER.

Heat-seeking missiles are dangerous. They sense the heat from your jet engine and home in on you. They'll catch you, too—they're faster than a jet.

Your assignment: Eliminate the heat-seeker base. It's easy enough to strafe the missiles on the ground, but if any are launched, you'll have to take evasive action.

If your Apple can display 80 columns of text, you *must* have it in 40-column mode before running the game. Press the Escape key, then Control-Q to enter 40-column mode.

Now type:

```
BRUN SEEKER
```

and hit Return. The game loads and runs, and you'll see the options menu screen. Either one or two people can play Heat Seeker. The two-player mode isn't competitive—the players take turns flying the plane, trying for the highest score. You need only one joystick, even with two players.

There are three levels of difficulty: Trainer, Novice, and Ace. All you need to do is press the T, N, or A key. In the Trainer level, missiles are never launched. You can fly as much as you like without being chased, but crashes are counted against your eight-jet total. This is the perfect level to become familiar with the controls of your aircraft. Novice and Ace are actual playing levels: Ace has faster action and tighter curves. All three levels award a flight-time bonus of ten points every few seconds, just for staying in the air.

Piloting The Jet

After you've selected your options, the game screen appears. Press the joystick fire button or the space bar to start your jet. Use the joystick or keyboard to control the movement of the plane. The controls may seem to be a little confusing at first. If you're using the joystick, pull back to loop upwards (counterclockwise) and push forward to loop down (clockwise), like a real airplane. The key controls are a bit different. The right-arrow key moves the jet downwards and the left-arrow key loops the aircraft upwards. The jet moves at a constant velocity—you can't speed up or slow down. Press either fire button on the joystick or the space bar to launch a missile at the heat seekers on the ground. It does no good to fire at a moving heat seeker. They're equipped with an Improved Electronic Evasion (IEE) circuit which makes them impossible to hit. The only way to get rid of a seeker is to make it crash into the ground. When you're being pursued, dive for the ground and pull up at the last second. Seekers are faster, but they can't turn as quickly.

Your plane can't shoot itself, so don't worry about all your shots zipping across the screen. If you manage to eliminate all the heat seekers, you get to start all over again, with a new group of heat seekers. A bonus of 1000 points is awarded for each group you eliminate. You have eight jets to work with—the number remaining is displayed on the screen, next to the score.

After crashing, you can catch your breath for a moment. You'll notice that all eight missiles have been replaced, even though you might have destroyed several with the previous jet. When you're ready to fly again, press the joystick button or the space bar to continue.

At the end of a game (after you've crashed eight jets), you can go back to the options menu to change selections, quit the game, or play the same game configuration again. (If you're using joystick controls, pressing the button gives you another play.)

Speedy Addiction

Heat Seeker is a commercial-quality, arcade-style game. With the exception of sound (omitted because of program length), Heat Seeker has all the elements you expect from something you've paid up to $20 for. It's fast, smooth, and graphically dazzling. More important, it's addicting. ©

Miami Ice

Jeff Kulczycki
Version by Tim Victor

Here's an action game that challenges both your driving skills and powers of concentration. A joystick is required. Disk File: MIAMI.ICE

Ah, Miami—sun city of the South. A sparkling metropolis blessed with a tropical climate, palm trees, beaches, revived art deco architecture, stylish pastels, and classy elegance. Almost paradise.

You wake up on another bright, sunny Miami morning, sip a glass of freshly squeezed orange juice, don your white linen suit and sunglasses, and stroll outside—then get the shock of your life.

What's going on here? Overnight, a freak shift in the jet streams has piped a blistering cold front down from Ohio. The weatherman had predicted a brief shower last evening, but that's not what happened. Instead, the Florida peninsula was blasted by the worst ice storm in 400 years. The Everglades are frozen solid. The pink flamingos are blue. And the streets of Miami are coated with a shimmering layer of slippery ice.

As you start your car—the pampered engine coughs and sputters in the bitter cold—you wonder what it's going to be like driving to work. A Miami native, you've never driven on ice before. In fact, you've never even *seen* this much ice since your boss's retirement party last year, when the caterers made that life-size ice sculpture of Ponce de Leon. You've heard the horror stories told by tourists about winter driving conditions up North, but never thought it could happen to you—not here, in Miami.

The minute you pull out onto the street, your worst fears come true. When you step on the gas pedal, the wheels spin and the car accelerates sluggishly. When you turn the steering wheel, the car slides all over the road. And when you step on the brakes—well, forget it.

You realize, desperately, that you've got to make it to the parking garage across town without smashing your car to smithereens. It won't be easy. But at least there's one thing in your favor—you've got the whole road to yourself. Everyone else, it seems, had the good sense to stay home.

Out Of Control

Using a joystick or paddles, you have to drive your car over ice-covered streets to reach the safety of a garage. The joystick button is the gas pedal, and pushing the stick right or left steers the car in the corresponding direction.

But here's the twist—the car doesn't respond instantly to your commands. It tends to slide in the same direction even after you've steered it toward another direction. Then, when you try to recover, you often overcorrect and start sliding in yet another new direction. It's an inertial nightmare—much like real winter driving.

When you hit a guardrail or some other obstruction, your car cracks up. You get three cars per game. If you reach the safety of the garage, the game isn't over. Instead, you advance to another screen whose streets are even harder to navigate.

The number of points you score depends on how soon you reach the garage. As an incentive to recklessness, a timer starts counting down when you begin each new screen. If you reach the garage, you score the number of points left on the timer. If the timer runs out, you can still reach the garage, but you won't get any points. However, you will advance to the next screen.

To begin playing, plug in a joystick or paddles. To reach the garage safely and advance to the next screen, you have to enter the front of the garage without bumping into any of its walls. There are seven screens in all. The game normally starts at screen 1, but you can begin a new game at any screen you want by pressing the controller button to change the screen number. ©

Hex War

Todd Heimarck, Assistant Editor
Verion By Tim Victor

You float high above a distant planet, controlling robot armies below. Can you take control of the priceless mining turf planetside, or will your opponent's robot crews prevail? To win at this thoughtfully designed, engaging strategy game, you'll need foresight and conceptual skills rather than a quick hand on the joystick. A color monitor or color TV is recommended. Disk File: HEXWAR.

"Hex War" is a two-player strategy game that can be played five different ways, and there are limitless variations. But the basic premise is always the same: You and an opponent move armies on a field of hexagons, attempting to capture territory.

The goal of the first two games is simple: capture the capital city of the other player. In game 1, the capital cities are far apart; you must devote some of your armies to defending your own capital while attempting to breach the walls of the other capital. Game 2 puts the capitals near each other, so offense and defense tend to merge in this scenario. Most of the action takes place within a small area of the battlefield.

Games 3 and 4 spread the action over a wider area. In the third game, your object is to occupy eight of the twelve cities on the game board. Six cities occupy the periphery, and six are in the center of the playfield. Game 4 requires actual control of six cities; you must have an army in the city, one that's not involved in a battle, before you're credited with control (this version will probably take the most amount of time to play).

Although the first four scenarios encourage a commitment to battle, you employ different tactics in the fifth. The goal here is to acquire 40 of the 61 hexes, so you need some free armies to move around. As soon as you claim 40 hexes, you win the game.

When you first run Hex War, the computer pauses to set up the screen, then displays a menu of five choices. The five different games are explained in detail below. If you're new to the game, press the 1 key to choose game 1. There will be another short pause while the variables are initialized, and then you'll see a playfield with 61 hex shapes, containing four armies on each side.

Hexes And Hexadecimal

A chess board has 64 squares arranged in a rectilinear grid. Hex War gives you a playing field of 61 hexagons (almost as many as a chess board), but they're part of a six-sided honeycomb field. If you've played war games before, you may recognize the hexes.

At first, the cursor movement may seem unusual. The cursor travels not up-down/left-right, but northeast-southeast/northwest-southwest. Use keyboard controls to maneuver the armies. Press the period (.) key to go northeast (use the > symbol on that key as a reminder), the comma (,) key to go northwest (note the < symbol on that key), the → to move southeast, and the ← key to move southwest. Use the space bar to select or set an army. Press RETURN to end your turn before all your armies have been moved.

Each hex has six neighbors, so an army can move in six possible directions.

Army strengths are listed in hexadecimal (base 16) numbers, so the four armies labeled 40 actually have strengths of 64 (the hexadecimal value 40 equals 64 in our everyday decimal numbering system). At the beginning of a turn, any army has exactly three movement points. It requires one point to move an army into a neutral or enemy-controlled zone. To move *through* the same zone also requires a point. To move into and through a friendly hex requires a total of one point. This means you can move a single army through two neutral or enemy hexes in any one turn, but the same army can move through up to three friendly zones during a turn.

Select an army by moving the cursor onto it and pressing the space bar. Then position the cursor on a neighboring hex and press the spacebar again. If you wish to stop, press Return, and two plus signs (++) will appear, signaling that no more movement can occur. Otherwise, position the cursor on another neighboring hex and click.

Zones Of Control

Each army controls the six contiguous hexes surrounding its resident hex. If you enter an enemy's zone of control, you forfeit any additional moves and must prepare for battle. In addition, an army that begins the turn in a zone of control cannot move until the battle is resolved.

Robots Vs. Robots

In this game, you aren't really on the planet, but parked high above it in a remote mothership. You've landed some robots to explore the area, and they've encountered robots belonging to another explorer. Your robots, or *bots* as you call them, follow your orders to advance toward the other bots. Each bot has a mining laser which can stop or disable the other bots. Also, your bots have disruptor beams which can daze another bot, temporarily confusing it. When two bot-groups come close to each other, they shoot lasers and disruptors until one army of bots is disabled.

Three things can happen to a robot which suffers a hit. If the robot suffers a direct hit in its logic

unit by a laser, it is vaporized. It is destroyed forever and never reappears in play.

The second thing that can happen is injury. If the laser beam is deflected, the robot is out of commission until it can be transported back to a botspital. An injured bot is frozen in place until the battle is finished, after which the victorious army carts away the injured bots to be repaired and reused.

Thus, winning a battle means you evacuate both the friendly injured and the enemy injured. After all of the injured bots recover, *they join the force in whose botspital they were healed.* In effect, injured bots eventually become members of the army which won the battle in which they were damaged.

The third possibility is confusion: The robot is temporarily disoriented for two turns. When the time has passed, the robot is ready again.

Reprogramming Bots

Moving the cursor onto an army of robots brings up a status window in the upper-left corner of the screen. The number in reverse video is unimportant; it's the army number (which may change as the game progresses).

The four numbers underneath are significant, however. The first is the army's active strength (in decimal). The second is the number of injured robots, which will be transported to the botspital of whichever side wins the battle. The third—on the line below—is the number of disrupted robots who will be available for combat in the next turn. The fourth number is how many robots can join the active force two turns from now.

If one side is able to reduce the other player's active force to zero, two things happen. The winner sends all injured bots away to be repaired. The winning side also collects all enemy bots (injured or dazed) and sends them to the reinforcement center to be reprogrammed. Eventually all these bots will available to the winner of this particular battle for future engagements.

Reinforcements And Mergers

At the start of the game, you'll see some armies positioned outside of the hex field. These are reinforcements and reserves in transit to the battle. Player one's reinforcements enter at the bottom right corner; player two's enter at the top left. The line of new armies moves counterclockwise; the army next to the entry point is the next to enter the battlefield.

However, the reinforcements cannot enter the battlefield if an army (friendly or enemy) is blocking their way. Keep your armies off your own reinforcement point, and try to block your opponent's armies from this area if you can. If the entry hex is owned but not occupied by your opponent, you'll lose some reinforcements.

After completing a turn, you are credited with additional reinforcements according to how much territory you own. Passing over a hex allows you to claim it; the hex changes color to indicate ownership. Each piece of property provides enough ore and energy to build a new robot, available for use two turns in the future. The numbers in the line of reinforcements are updated after you move to show additional robots being built.

Winning a battle also provides additional armies in the line of reinforcements. As mentioned above, a victorious army captures any dazed enemy bots, which are reprogrammed and available in three turns. At the same time, the winner evacuates injured bots of both sides. Transportation and repair take five turns for friendly bots, seven for enemy bots. The two additional turns are needed for reprogramming the opponent's forces.

If you're losing a battle, the number of injured robots (displayed in the status window) will begin to rise. Remember that, if your opponent reduces your active strength to zero, he or she will capture all of your injured bots; they'll be reprogrammed and added to future reinforcements. To prevent this from happening, you're allowed to bring in a second army for merging. Simply move another army on top of the army with which you want to merge. There's just one rule: One or both of the armies must have a strength less than 32 decimal (1F or less in hex). ©

Apple Bowling Champ

Original Program By Joseph Ganci
Version By Patrick Parrish, Programming Supervisor

Now you can go bowling without the expense of renting special shoes or suffering the embarrassment of rolling a gutter ball in front of dozens of people. "Bowling Champ" is a game for one to four players which runs on any Apple II-series computer. Disk File: BOWLING.CHAMP.

Some computer games, such as *Pac-Man* or *Adventure*, create their own unique fantasy worlds, while others are simulations of reality. "Apple Bowling Champ" is an example of the latter.

It's not easy to take a game with countless physical variables such as bowling and reduce it to numbers so it can be re-created by a computer—especially a microcomputer. Compromises must be made. Usually the game must be modified in major ways to make it possible to program. The result is a hybrid game, an approximation of reality, that resembles the original but has new aspects of its own.

Apple Bowling Champ is a reasonable simulation of a game of tenpins, given the limitations imposed by a BASIC program which must remain short enough to publish in a magazine. The elements of skill and luck have been preserved, and the scoring is authentic.

Up To Four Players

When you run Bowling Champ, the program asks for the number of players. Up to four people can play. Next, enter the players' names. To fit the names on the 40-column screen, the program truncates entries to eight characters.

Now you're ready to bowl the first frame. The bowling ball moves rapidly up and down across the alley until you press the space bar. This rolls the ball down the alley and knocks over the pins—unless you've thrown a gutter ball. The trick is to time your release so the ball rolls down the center of the alley to score a strike.

In case you're unfamiliar with how a game of tenpins is scored, here's a brief summary.

A game consists of ten frames or turns. Each player gets one or two balls per frame. If you roll a strike—knocking down all tenpins with your first ball—you don't get a second ball, but the current ball's score is ten plus the total of your next two throws.

If some pins are left standing after your first ball, you get a second ball. If you knock down all the remaining pins, it counts as a spare, and the current ball's score is ten plus your next throw.

If any pins remain after your second ball (no strike or spare), the number of pins knocked down in that frame is added to your previous score.

Rolling a spare in the tenth (last) frame gains you one extra ball; rolling a strike in the tenth frame gains two extra balls.

Therefore, a perfect game—ten strikes during regular play plus two strikes with the extra balls—scores 300 points. Needless to say, this doesn't happen very often, either in real bowling or in Bowling Champ.

Since Bowling Champ follows every rule of scoring for regular bowling, you can learn how to score by carefully observing the game. The only difference is that the computer does not wait until the end of a frame to update the score; it updates it after every ball.

Adjusting The Difficulty

Novice bowlers may find that the ball moves too fast for them to aim. On the other hand, more experienced players may want to speed up the ball to make the game harder. You can easily make either modification by changing the delay loop in line 480. The statement in 480 reads:

```
480 FOR R=1 TO 10:NEXT
```

Replacing the 10 with a larger number slows down the ball; a smaller number speeds up the ball. You might try a value between 20 and 50 for youngsters. For expert players, remove line 480 altogether. ©

Tug-A-War

Mark Tuttle
Version by Tim Victor

Don't be fooled by the apparent simplicity of this two-player strategy game. It looks easy on the surface, but it's a stiff test of your concentration and ability to think ahead. Since the games is based on colors, a color monitor or TV is required. Disk File: TUGAWAR.

Nearly everyone has played tug of war at one time or another. The traditional game pits two players or teams at opposite ends of a rope. At the middle of the rope is a flag, and each side tries to pull the flag into its territory. "Tug-a-War" is based on a similar concept. In this version, the flag is replaced with a round ball shape, and each player tries to maneuver the ball onto his or her side of the screen. Like many two-player games, the difficulty of Tug-a-War depends somewhat on the intelligence of your opponent. But even at the simplest level, you'll find that skill and foresight are essential to success.

Battle Of The Colors

When you run Tug-a-War, two sets of colored boxes appear, one above the other. The lower, longer series of squares is the playing field. Near the middle of the playfield area is a round ball; the outermost boxes at each end of the playfield represent each player's home position. The players alternate turns, each trying to move the ball in their own direction, until it reaches one of the home squares.

So far, so good—but how do you move the ball? It's done not by pulling a rope, but by changing the colors of boxes in the playfield. The color of the square under the ball determines which direction it moves and how far it travels. On any given turn, the ball can move either one or two squares to the left, or one or two squares to the right.

At the top of the screen are four boxes that show you which colors are linked to which directions. For instance, the leftmost box shows you which color makes the ball move one square to the left. The next box to the right shows you which color makes it move *two* squares to the left. The second pair of boxes show you which colors make the ball move in the opposite direction, to the right. By changing the color of the box where the ball is currently located, you can make it move toward your home square.

The playfield contains 11 boxes. When the game begins, each of these boxes is randomly given one of the four colors shown at the top of the screen. On each turn, you may change the color of one, several, or all of the boxes (however, you must always change at least one box). Below each box is a number which represents its distance from the home position of the player whose turn it is. For instance, if you are the player on the left, then on your turn the boxes are numbered 1, 2, 3, etc., from left to right (the tenth box is marked with a 0, and the eleventh with an A). When it's the right player's turn, the numbering is reversed (the rightmost box is 1, etc.).

To take a turn, you must select a number that corresponds to the numbers shown below the boxes in the playfield. This is done by pressing a single key. Press a number key from 1–0 to select one of the first ten values, or press the A key to choose the eleventh box. The number you choose determines how many boxes change color. For instance, if you press 1, only one box (the one nearest your home square) changes color. If you press 2, the two boxes nearest your home box change, and so on.

Where do the new colors come from? Every box cycles through the same series of four colors shown in the uppermost set of boxes, going from left to right. For example, if the colors shown there are white-blue-red-purple (the exact colors may be different on your computer), then a white square always changes to blue; a blue square always changes to red; a purple square changes to white, and so on. In other words, the box's current color determines which color it gets after the next color change.

Though every turn involves at least one color change, the ball doesn't necessarily move on every turn. It only moves when you change all the boxes between your home position and the current position of the ball. For example, if the ball is three boxes away from your home square, then you must change the color of at least three boxes in order to move it at all.

Foresight Rewarded

As you can see, there's much more to this game than appears on the surface. At first you might be tempted to try to move the ball as often as possible. But that's usually a losing strategy. Remember, the *direction* the ball moves depends on the color of its square before you take the turn.

In many cases, you'll want to move the ball only if it's on a color that moves it toward your goal. But like other games of strategy, Tug-a-War rewards the player who looks beyond the current move and tries to set things up for future moves; sometimes it's wise to make a small, temporary sacrifice in order to benefit later in the game. Because the boxes change colors in the same sequence, the effect of your own move is always completely predictable. However, since a single turn can change the color of many boxes, dramatic changes of fortune are also possible. ©

Balloon Crazy

Joseph Russ

Catch as many balloons as you can—but be careful not to fall off your skateboard. The game can be played with paddles. Disk File: BALLOON.

"Balloon Crazy" is a game that children can enjoy, yet its higher levels are a challenge for adults. The goal is simple: You must zip back and forth across the screen on a skateboard while catching falling balloons on top of your head. Since some of the balloons fall very fast, that's not as easy as it sounds. After you've caught enough balloons, you can reach up to pop them, then catch some more. If you miss just one, you lose all the balloons currently in your possession.

Oodles Of Balloons

Each game begins by displaying several rows of multicolored balloons at the top of the screen. You are the skateboarder at the bottom. When a balloon begins to fall, move directly under it and catch it on your head. The blue balloons fall slowly, which makes them easy to catch (but worth the fewest points). The green balloons fall faster, but swiftest of all are the red balloons. As soon as you snare a balloon, it joins the pile on top of your head.

Points are scored only when you have caught the required number of balloons. You have three players to work with in each game: Falling off the skateboard costs you one player. Clearing all the balloons from a screen permits you to advance to the next level—where everything becomes more difficult. Bonuses are awarded at appropriate intervals, and you can earn an extra player by scoring 1,000 points.

Though the balloons become harder to catch at higher levels, you are never helpless to prevent them from hitting the ground. Should a balloon miss the top of the pile, you can bounce it back into the air by pressing the paddle button or the space bar and running into it. The balloon will then float back into the air, and you may try to catch it again.

To play the game type:

```
BRUN BALLOON
```

You can play the game with a paddle on any Apple II computer: Move the paddle to control the player, and press the paddle button to bounce the balloon upward. Alternatively, keyboard controls can be used on the Apple IIc and Apple IIe: press the open-Apple key to move left, the closed-Apple key to move right, and the space bar to bounce.

Four balloons must be collected to score points. If you miss a balloon completely, all the balloons on your head drift off into space and disappear. There are nine game levels. Red balloons do not appear until the second level, but each higher level contains more red balloons. You may pause the game by pressing RETURN; resume play by pressing the space bar. ©

Hickory, Dickory, Dock

Barbara H. Schulak

Version by Tim Victor, Editorial Programmer

This fun, educational program helps children learn the concepts of telling time by relating a digital clock display to a conventional clock face. Disk File: HICKORY.

"Hickory, Dickory, Dock" offers an enjoyable way for children to learn how to tell time. When you run the program, it displays a round clock face as well as a digital display. Four different activities are available. The first option lets youngsters practice telling time. As the positions of the clock hands change on the screen, the digital clock display changes as well. This shows the relationship between the spatial position of hands on a clock face and the numeric representation of time.

The other three activities test a youngster's time-telling ability for hours only, hours and half-hours, or five-minute intervals. Move the hands to the correct position, then press RETURN to enter the answer. Press 1 to move the minutes hand; press 2 to move the hours hand. After five correct answers, the program plays a brief song and displays some graphics as a reward. After three incorrect choices, the program automatically moves the clock hands to the correct position. ©

Apple SpeedScript 80

The 80-Column Machine Language Word Processor For Apple II

Charles Brannon, Kevin Martin, and Bill Rodrick

Apple SpeedScript 80 *is an 80-column word processor for Apple IIe, IIc, and IIGS computers. Based on* SpeedScript 3.0, *which was first published in the June 1985 issue of COMPUTE! magazine,* Apple SpeedScript 80 *is an impressive word processing application that's easy to learn, and just as easy to use.*

Apple SpeedScript 80 *runs on any Apple IIc or IIGS, and on IIe computers with an 80-column card. DOS 3.3 and ProDOS versions are included. Disk Files: SS80.SYSTEM, SS3.3.SYSTEM, CONVERTER, CONVERTER3.3*

SpeedScript 80, though compact in size (5.5K), has most of the functions you expect in a full-featured word processor. *SpeedScript* is also very easy to learn and use. You type in everything first; preview and make corrections on the screen; insert and delete words, sentences, and paragraphs; then print out an error-free draft, letting *SpeedScript 80* take care of things like margins, centering, headers, and footers.

Running *SpeedScript 80*

To run *SpeedScript 80* with DOS 3.3, reboot the machine to clear out memory. Then, from BASIC, type **BRUN SS3.3.SYSTEM.** *SpeedScript 80* automatically loads and runs. If you prefer, you can write a short program, something like

```
10 PRINT CHR$(4);"BRUN
   SS3.3.SYSTEM"
```

Save this program as the HELLO file on the disk (use INIT HELLO to format and set up a blank disk). This makes *SpeedScript 80* load and run automatically when you boot up.

To run the ProDOS version, simply reboot the machine with the *SpeedScript 80* (SS80.SYSTEM) startup disk in the drive, and *SpeedScript 80* will load and run automatically. Alternatively, if you already have ProDOS booted, insert the *SpeedScript 80* disk and at a BASIC] prompt, enter:

```
-SS80.SYSTEM
```

An Empty Page

When you run *SpeedScript 80,* you'll see a blank 80-column screen with a blinking underline cursor. The first line on the screen is in reverse video, white with black letters. *SpeedScript 80* presents all messages on this *command line.* The remaining 20 lines of the screen are used to enter, edit, and display your document.

The clarity of the 80-column display will vary depending on your monitor or television set. If the characters appear fuzzy and a bit unreadable, try adjusting the contrast of the monitor or set while turning up its brightness. This leaves you with a gray background instead of black, but the characters should show up more easily.

The cursor shows where the next character you type will appear on the screen. *SpeedScript 80* lets you move the cursor anywhere within your document, making it simple to find and correct errors.

Entering Text

To begin using *SpeedScript 80,* start typing, just as you would on a typewriter. Hold down the Shift key to type an uppercase letter, and be sure the Caps Lock key is disengaged if you want to type lowercase.

When the cursor reaches the right edge of the screen, it automatically jumps to the beginning of the next line, just as in BASIC. But unlike BASIC, *SpeedScript 80* never splits words at the right edge of the screen. If a word you're typing won't fit at the end of one line, it's instantly moved to the next line. This feature, called *word-wrap*, helps make your text more readable.

Scrolling And Screen Formatting

When you finish typing on the last screen line, *SpeedScript 80* automatically scrolls the text upward to make room for a new line at the bottom. Imagine the screen as a 20-line window on a long continuous document. The DOS 3.3 version provides room in memory for 27,904 characters, or about 10–15 pages of text. Unfortunately, the DOS 3.3 version cannot make use of memory in excess of 48K. However, the ProDOS version of *SpeedScript 80* does give you access to an additional 10K, or over 10,000 more characters.

To check at any time how much space is available, press **Control-A** (hold down the Control key while pressing the A key). The number appearing in the command line indicates how much *available* room remains for characters of text.

If you're used to a typewriter and this is your first experience with word processing, you'll have to unlearn some habits. For one thing, you don't have to press Return at the end of each line as you do on a typewriter. *SpeedScript 80*'s word-wrap takes care of this automatically. Press Return only when you want to force a carriage return to end a paragraph or limit the length of a line. A *return mark* appears on the screen as a reverse less-than sign (<).

Using The Keyboard

Most features are accessed with Control-key commands—you hold down Control while pressing another key. In this article, Control-key commands are represented as **Control-*x*** (where *x* is the key you press in combination with Control). An example is the Control-A, mentioned above, to check on available memory. Control-E means hold down Control and press E.

Some commands have special options. You may be required to press the Open Apple key (the key to the left of the space bar with a hollow Apple symbol on the face) while pressing another key or Control-key combination. The Open Apple key is abbreviated as **OpAp** in this article. **OpAp-Control-*x*** means to press Open Apple in conjunction with a Control-key combination. Other keys are referenced by name or function. See the "Keyboard Map" figure for a complete quick-reference chart of all keyboard commands. Because *SpeedScript 80* uses almost every key, not all combinations are especially mnemonic. Most keys, though, stand for the name of the function they perform.

Some keys let you move the cursor to different places in the document to make corrections or scroll text into view. You can move the cursor by character, word, sentence, or paragraph. Here's how to control the cursor:

- The ← and → keys (cursor-left and cursor-right) move the cursor a single space in either direction. By holding down the Open Apple key while pressing the appropriate cursor key, you can move the cursor to the beginning of the next (→) or previous (←) word.
- The ↑ and ↓ keys (cursor-up and cursor-down) move the cursor to the beginning of either the previous or next sentence. A sentence is defined as any sequence of characters ending in a period, question mark, exclamation point, or return mark. Hold down the Open Apple key as you press the arrow to move to the beginning of the next (↓) or previous (↑) paragraph. A paragraph is defined as any sequence of characters ending in a return mark.
- Pressing **Control-@** (Control-Shift-2) puts the cursor at the top of the screen. If the cursor is already at the top of the screen, Control-@ moves the cursor to the top of the document. To move quickly to the beginning of the document, press Control-@ twice.
- **Control-Z** moves the cursor to the end of the document, scrolling if necessary. It's easy to remember since Z is at the *end* of the alphabet.

Making Corrections

Sometimes, you'll have to insert characters to make a correction. Use **Control-O** to *open up* a single space. Merely position the cursor at the point where you want to insert a space, and press Control-O.

It can be tedious to use Control-O to open up enough space for a whole sentence or paragraph. For convenience, *SpeedScript 80* has an insert mode that automatically inserts space for each character you type. In this mode, you can't type over characters; everything is inserted at the cursor position. To enter insert mode, press **Control-I.** To cancel insert mode, press Control-I again. To let you know you're in insert mode, the cursor changes from a blinking underline to a blinking reverse underline, which looks like a solid square. The cursor changes back to a blinking underline when you exit insert mode. Because of keyboard decoding redundancy, the Tab key works just like Control-I.

Insert mode is the easiest way to insert text, but it can become too slow when you're inserting near the top of a very long document, because it must move all the text following the cursor position. So *SpeedScript 80* has even more ways to insert blocks of text.

One way is to use **Control-T**. It's programmed in *SpeedScript 80* to act as a five-space margin indent. To end one paragraph and start another, press Return twice and press Control-T. A shortcut for this is **Control-]**. This key combination automatically inserts two return marks and then indents the margin. Control-T always inserts; you don't need to be in insert mode. You can also use Control-T to open up more space than Control-O. (You cannot set or clear tab stops in *SpeedScript 80* as you can with some word processors.) No matter how much space you want to insert, each insertion takes the same amount of time. So Control-T can insert five spaces five times faster than pressing Control-O five times.

There's an even better way, though. Press **Control-Q** to *quickly* insert 255 spaces. (It doesn't insert a line; use Return for that.) You can press it several times to open up as much space as you need. Control-Q

inserts 255 spaces as fast as Control-O opens up one space. Now just type the text you wanted to insert over the blank space. (You don't want to be in Control-I insert mode when you use this trick; that would defeat its purpose.)

Since Delete is also slow when you're working with large documents (it, too, must move all text following the cursor), you may prefer to use the ← key to backspace when using this method.

After you've finished inserting, there may be some inserted spaces left over that you didn't use. Just press **OpAp-Control-Q**. This instantly deletes all extra spaces between the cursor and the start of following text. It's also useful whenever you need to delete a block of spaces for some reason.

Erasing Text

To erase the character on which the cursor is sitting, press **Control-G** (to *grab* the character). The character highlighted by the cursor is removed, and all following text is moved toward the cursor to fill the empty space.

Press the **Delete** key to delete the character to the left of the cursor. All the following text is moved with the cursor to fill the empty space.

These keys are fine for minor deletions, but it could take all day to delete a whole paragraph using them. So *SpeedScript 80* has two commands that can delete an entire word, sentence, or paragraph at a time. **Control-E** erases text *after* (to the right of) the cursor position (and can also erase all text), and **Control-D** deletes text *preceding* the cursor.

To use the Control-E erase mode, first place the cursor at the beginning of the word, sentence, or paragraph you want to erase. Then press Control-E. The command line shows the message ERASE (S,W, P,A): RETURN TO EXIT. Press S to erase a sentence, W for a word, or P for a paragraph. Each time you press one of these letters, the text is quickly erased. You can keep pressing S, W, or P until you've erased all the text you wish. Then press RETURN to exit the erase mode.

You also use Control-E to erase all text from memory when you want to start a new document. To erase all text, press Control-E; then press the A *(all)* key. A prompt appears: ERASE ALL TEXT: ARE YOU SURE (Y/N). Press Y to perform the irreversible deed. You cannot recover any text erased in this way. Press N or any other key to cancel this function.

The Control-D delete mode works similarly to Control-E, but it deletes only one word, sentence, or paragraph at a time. First, place the cursor after the word, sentence, or paragraph you want to delete. Then press Control-D. Next, press S, W, or P for sentence, word, or paragraph. The text is immediately deleted and you return to editing. You don't need to press Return to exit the Control-D delete mode unless you pressed this key by mistake. *(In general, you can escape from any function in SpeedScript 80 by simply pressing Return.)* Control-D is most convenient when the cursor is already past what you've been typing.

The Text Buffer

When you erase or delete with Control-E and Control-D, the text isn't lost forever (unless you've performed an Erase All). *SpeedScript 80* remembers what you've removed by storing deletions in a separate area of memory called a *buffer*. The buffer is a fail-safe device. If you erase too much or change your mind, just press **Control-^** (Control-Shift-6) to restore the deletion. Be aware, however, that *SpeedScript 80* normally remembers only the last erase or delete you performed. It's not hard to remember this command, since the ^ (caret) symbol is used in paper-and-pencil editing to signify an insertion.

Another, more powerful use of this buffer is to move or copy sections of text. To move some text from one location in your document to another, first erase or delete it with Control-E or Control-D. Then move the cursor to where you want the text to appear and press Control-^. Control-^ instantly inserts the contents of the buffer at the cursor position. If you want to copy some text from one part of your document to another, just erase or delete it with Control-E or Control-D, restore it at the original position with Control-^, then move the cursor elsewhere, and press Control-^ to restore it again in the new position. You can retrieve the buffer with Control-^ as many times as you like. If there is no room left in memory for inserting the buffer contents, you'll see the message NO ROOM.

Important: The Control-E erase mode lets you erase up to the maximum size of the buffer (2K, or about 2,000 characters). If you try to erase more text than the buffer will hold, you'll get the message BUFFER FULL, and erase mode will be canceled. Each new Control-E removes the previous contents of the buffer. Keep this in mind if there's something in the buffer you'd rather keep. If you don't want the current buffer contents to be erased, hold down the Open Apple key when you press Control-E. **OpAp-Control-E** works just like Control-E, except that it preserves the current buffer contents and adds the newly erased text to the buffer.

If you ever need to erase the contents of the buffer, press **Control-N** *(null buffer)*. As an added safety measure, you'll be asked if you are sure you want to erase the buffer. Press Y to null the buffer and N (or any other key except Y) to cancel the command.

Search And Replace

SpeedScript 80 has a FIND command that searches through your document to find a selected word or phrase. A Replace option lets you automatically change one word to another throughout the document.

OpAp-Control-F *(find)* lets you define a search phrase, or **OpAp-Control-R** *(replace)* lets you define a replace phrase, and **Control-Y** is used for automatic searching and replacing.

Searching is a two-step process. First, you need to tell *SpeedScript 80* what to search for; then you trigger the actual search. Hold down Open Apple and press Control-F. The command line prompts FIND:. Type in what you'd like to search for—the *search phrase*—and then press Return. If you press Return alone without typing anything, the Find command is canceled.

When you're ready to search for the specified phrase, press **Con-**

trol-F (without the Open Apple). *SpeedScript 80* looks for the next occurrence of the search phrase *starting from the current cursor position.* If you want to hunt through the entire document, press Control-@ twice to move the cursor to the very top before beginning the search. Each time you press Control-F, *SpeedScript 80* looks for the next occurrence of the search phrase and places the cursor at the start of the phrase. If the search fails, you'll see the message NOT FOUND.

Control-R works together with Control-F. After you've specified the search phrase with OpAp-Control-F, press OpAp-Control-R to specify the replace phrase. (You can press Return alone at the REPLACE WITH: prompt to select a *null* replace phrase. When you hunt and replace, this deletes the located phrase.) To search and replace manually, start by pressing Control-F. After *SpeedScript 80* finds the search phrase, press Control-R if you want to replace the phrase. If you don't want to replace the phrase, don't press Control-R. You're not in a special search-and-replace mode. You're free to continue writing at any time.

Control-Y links Control-F and Control-R together. (Think of the two branches of the Y linking together Find and Replace.) It first asks for a search phrase, then asks for a replace phrase, and finally, automatically searches and replaces throughout the document starting from the current cursor position.

There are a few things to watch out for when you're using search and replace. First, realize that if you search for *the, SpeedScript 80* finds the embedded *the* in words like *therefore* and *heathen.* If you changed all occurrences of *the* to *cow,* these words would become *cowrefore* and *heacown.* If you want to find a single word, be sure to include a space as the first character of the word, since almost all words are preceded by a space. Naturally, if you're replacing, you need to include the space in the replace phrase, too.

SpeedScript 80 also distinguishes between uppercase and lowercase. The word *Meldids* does not match with *meldids. SpeedScript 80* will not find a capitalized word unless you capitalize it in the search phrase. To cover all bases, you will sometimes need to make two passes at replacing a word. Keep these things in mind when using Control-Y, since you don't have a chance to stop a linked find-and-replace.

Storing Your Document

To store your text, press **Control-S.** You'll see the prompt SAVE:. Type in a legal filename and press Return. *SpeedScript 80* defaults to drive 1. If you need to access a second disk drive, precede the filename with **2:**. Once specified, this becomes the default drive for future disk access. To return to drive 1, precede the filename with **1:**.

If the filename you specify coincides with one already on the disk, the existing file will be overwritten by the new one. (You won't be given any warning, so be careful not to accidentally use duplicate filenames.) You can press **Control-C** to display a catalog of the files on the disk. The catalog pauses when the screen is full, waiting for you to press a key to continue. When the catalog is finished, press Return to return to editing.

Control-S always saves the entire document. The cursor position within the document is not important.

When the save is complete, *SpeedScript 80* reports NO ERRORS if all is well, or it gives a message like DISK FULL if not. Check your DOS or BASIC manual for a list of error messages and their causes.

Loading A Document

To recall a previously saved document, press **Control-L.** Answer the LOAD: prompt with the filename. Again, you can precede the filename with **1:** or **2:** to switch drives as necessary. *SpeedScript 80* loads the file and will display NO ERRORS if no problems are encountered in the load; otherwise, it reports the error.

It's important to position the cursor correctly before loading a file. Documents start loading at the cursor position, so be sure to press Control-@ twice or Control-E and A (Erase All) to move the cursor to the start of text, unless you want to merge two documents. When you press Control-L to load, a flashing asterisk appears to warn you if the cursor is not at the top of the document.

To merge two or more files, simply load the first file, press Control-Z to move the cursor to the end of the document, and then load the file you want to merge. Do not place the cursor somewhere in the middle of your document before loading. A load does not insert the text from disk, but overwrites all text after the cursor position. The last character loaded becomes the new end-of-text pointer, and you cannot access any text that appears ahead of this pointer.

SpeedScript 80 stores documents as binary files, so you cannot read a document in BASIC, load a BASIC program into *SpeedScript 80* for editing, or upload a document file with a modem. However, CONVERTER and CONVERTER3.3 are file-conversion programs which allow these functions. They convert *SpeedScript 80* binary document files into ASCII text files. They can also convert an ASCII text file into a *SpeedScript 80* binary file. This lets you convert documents from other word processors or change a *SpeedScript 80* document into a text file suitable for uploading via telecommunications. You can even edit BASIC programs with *SpeedScript 80.* Add the following line to your BASIC program and run it. It creates a text file of your BASIC listing.

```
0 PRINT CHR$(4);"OPEN filename":
  PRINT CHR$(4)"WRITE filename":
  LIST
```

Use the appropriate File Converter program to convert the program text file into a *SpeedScript 80* file. You can then load this file into *SpeedScript 80* for editing. Save the edited file back to disk, run the File Converter program again to convert the *SpeedScript 80* file back to a text file, and then use EXEC *filename* to read the text file back into BASIC. Although this seems rather tedious, you may find it worthwhile when working with long programs. A similar technique can be used to edit files written by other applications.

Note: Delete any print-formatting commands (described below) before using the File Converter program to change a *SpeedScript 80* file into a text file. Otherwise, the em-

bedded commands will not be converted properly, and the text file may not be usable.

Additional Features

SpeedScript 80 has a few commands that don't do much, but are nice to have. **Control-X** exchanges the character under the cursor with the character to the right of the cursor. Thus, you can fix transposition errors with a single keystroke. **Control-W** (think *sWitch)* changes the character under the cursor from uppercase to lowercase, or vice versa.

Apple *SpeedScript 80* traps the Reset key. Reset and Control-Reset always return you to editing mode. There's no way to exit *SpeedScript 80* once you've run it, short of rebooting.

Printing

If you already think *SpeedScript 80* has plenty of commands, wait until you see what the printing package offers. *SpeedScript 80* supports an array of powerful formatting features. It automatically fits your text between left and right margins that you can specify. You can center a line or block it against the right margin. *SpeedScript 80* skips over the perforation on continuous-form paper, or it waits for you to insert single-sheet paper. A line of text can be printed at the top of each page (a *header*) and/or at the bottom of each page (a *footer*), and can include automatic page numbering, starting with whatever number you like.

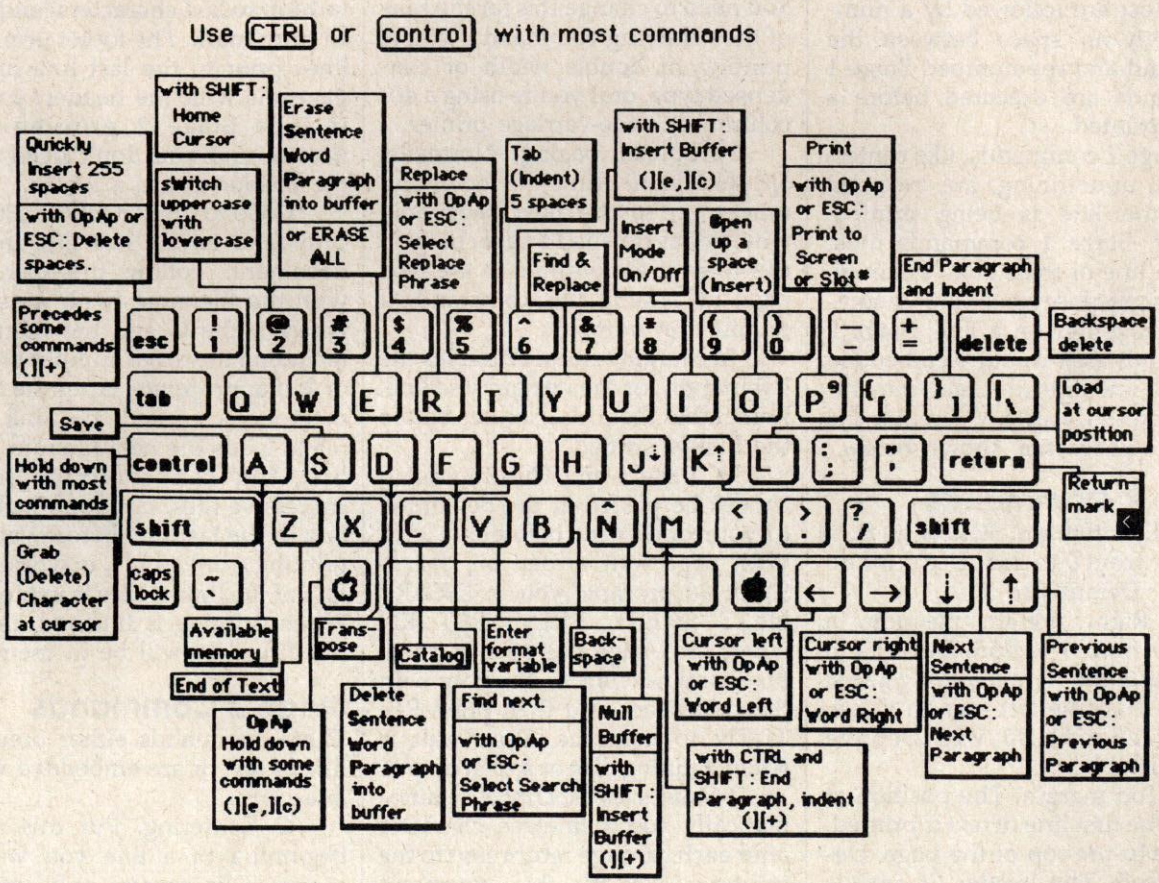
SpeedScript 80 can print on different lengths and widths of paper, and single-, double-, or triple- spacing (or any spacing, for that matter) is easy. You can print a document as big as will fit on a disk by linking several files together during printing. You can print to the screen or to a file instead of to a printer. You can send special codes to the printer to control features like underline, boldface, italics, and double-width typeface (depending on the printer).

But with all this power comes the need to learn additional commands. Fortunately, *SpeedScript 80* sets most of these variables to a *default* state. If you don't change the settings, *SpeedScript 80* assumes a left margin position of 5, a right margin position of 75, no header or footer, single-spacing, and continuous-paper page feeding. You can change these default settings if you want. (See below.)

Before printing, be sure the paper in your printer is adjusted to top-of-form (move the paper perforation just above the printing element). One additional note: Some printers incorporate an automatic skip-over-perforation feature. The printer skips to the next page when it reaches the bottom of a page. Since *SpeedScript 80* already controls paper feeding, you need to turn off this automatic skip-over-perforation feature before running *SpeedScript 80,* or paging won't work properly.

To begin printing, press **Control-P.** If your printer is attached, powered on, and selected (online), *SpeedScript 80* begins printing immediately. To cancel printing, press Control and the Reset key.

If you need to print to an RS-232 printer or to a printer in a slot other than slot 1, press **OpAp-Con-**



trol-P. This brings up the prompt PRINT TO: SCREEN, PRINTER? Press S to print to the screen. *SpeedScript 80* automatically prints to the screen, simulating the printer.

If you select P, you'll be asked for the slot number. Output is sent to the slot number you select. It's similar to PR# in BASIC. If you print to slot 6 (the disk drive), you'll cause the system to reboot, so be careful.

Formatting Commands

The print formatting commands are single letters embedded in text, such as L for left margin. To enter a formatting variable, press **Control-V**. You'll see the prompt ENTER FORMAT VARIABLE:. Now press any key. The print formatting commands must be distinguished from normal text, so they appear on-screen in reverse video. All lettered printer commands should be entered without the Shift key. During printing, *SpeedScript 80* treats these characters as printing commands.

There are two kinds of printing commands—Stage 1 and Stage 2. Stage 1 commands usually control such variables as left and right margins. Most are followed by a number, with no space between the command and the number. Stage 1 commands are executed before a line is printed.

Stage 2 commands, like centering and underlining, are executed while the line is being printed. Usually, Stage 1 commands must be on a line of their own, although you can group several Stage 1 commands together on a line. Stage 2 commands are by nature embedded within a line of text. Again, remember to press Control-V to enter the boldface characters shown below.

Stage 1 Commands

L Left margin. Follow with a number from 0 to 255. Use 0 for no margin. Defaults to 5.

R Right margin position, a number from 1 to 255. Defaults to 75. Be sure the right margin value is greater than the left margin value, or *SpeedScript 80* will behave unpredictably.

T Top margin. The position at which the first line of text is printed, relative to the top of the page. Defaults to 5. The header (if any) is always printed on the first line of the page, before the first line of text.

B Bottom margin. The line at which printing stops before continuing to the next page. Standard 8½ × 11–inch paper has 66 lines. Bottom margin defaults to line 58. Don't make the bottom margin greater than the page length.

P Page length. Defaults to 66. If your printer does not print six lines per inch, multiply lines-per-inch by 11 to get the page length. European paper is usually longer than American paper—11⅝ or 12 inches. Try a page length of 69 or 72.

S Spacing. Defaults to single-spacing. Follow with a number from 1 to 255. Use 1 for single-spacing, 2 for double-spacing, and 3 for triple-spacing.

@ Start numbering *at* page number given. Page numbering normally starts with 1.

? Disables printing until selected page number is reached. For example, a value of 3 would start printing the third page of your document. Normally, *SpeedScript 80* starts printing with the first page.

X Sets the page width in columns (think *a cross*). Defaults to 80. You need to change this for the sake of the centering command if you're printing in double width or condensed type, or if you're using a 40-column or wide-carriage printer.

N Forced paging. Normally, *SpeedScript 80* prints the footer and moves on to the next page only when it has finished a page, but you can force it to continue to the next page by issuing this command. It requires no numbers.

M Margin release. Disables the left margin for the next printed line. Remember that this executes before the line is printed.

W Page wait. This command should be placed at the beginning of your document before any text. With page wait turned on, *SpeedScript 80* prompts you to INSERT NEXT SHEET, PRESS RETURN when each page has been printed. Insert the next sheet, line it up with the printhead, and then press RETURN to continue. Page wait is ignored during disk or screen output.

J Causes *SpeedScript 80* to automatically add a linefeed character after each carriage return sent to the printer. Like **W**, this command must be placed before any text. Use this command only if your printer puts all text on the same line when you try to print your document. (Don't use this command to achieve double-spacing.)

I Information. This works like REM in BASIC. You follow the command with a line of text, up to 255 characters, ending in a return mark. This line will be ignored during printing and is handy for making notes to yourself (such as the filename of the document).

H Header define and enable. The header must be a single line of text (up to 254 characters) ending in a return mark. The header prints on the first line of each page. You can include Stage 2 commands such as centering and page numbering in a header. You can use a header by itself without a footer. The header and footer should be defined at the top of your document, before any text. If you want to prevent the header from printing on the first page, put a return mark by itself at the top of your document before the header definition.

F Footer define and enable. The footer must be a single line of text (up to 254 characters) ending in a return mark. The footer prints two lines prior to the last line of each page. As with the header, you can include Stage 2 printing commands, and you don't need to set the header to use a footer.

G Go to (link) next file. Put this command as the last line in your document. Follow the command with the filename (with no spaces between the **G** and the filename), including the drive number prefix **1:** or **2:** if appropriate. After the text in memory is printed, the link command loads the next file into memory. You can continue linking successive files, but don't include a link in the last file. Before you start printing a linked file, make sure the first of the linked files is in memory. When printing is finished, the last file linked to will be in memory.

Stage 2 Commands

These commands either precede a line of text or are embedded within one.

C Centering. Put this at the beginning of a line you want to center. This centers only one line

ending in a return mark. Repeat this command at the beginning of every line you want centered. Centering uses the page-width setting (see above) to center the line properly. To center a double-width line, either set the page width to 40 or pad out the rest of the line with an equal number of spaces. If you use double width, remember that the spaces preceding the centered text will be double-wide spaces.

E Edge right. This works in the same way as centering, but it blocks the line flush with the right margin.

When *SpeedScript 80* encounters this command, it prints the current page number. You usually embed this within a header or footer.

U A simple form of underlining. It works only on printers that recognize CHR$(8) as a backspace and CHR$(95) as an underline character. Underlining works on spaces, too. Use the first **U** to start underlining and another one to turn off underlining.

Fonts And Styles

Most dot-matrix printers are capable of more than just printing text at 10 characters per inch. Some printers have several character sets, with italics and foreign language characters. Most can print in double width (40 characters per line) or condensed (132 characters per line), and in either pica or elite. Other features may include programmable characters, programmable tab stops, and graphics modes. Many word processors customize themselves to a particular printer, but for flexibility *SpeedScript 80* was purposely designed not to be printer-specific. Instead, *SpeedScript 80* lets you define your own Stage 2 printing commands.

You define a programmable *printkey* by choosing any character that is not already used for other printer commands. The numbers 0–9, most symbols, and some alphabetic characters are available for printkeys. You enter these commands, like printer commands, with Control-V. The printkeys are like variables in BASIC.

To define a printkey, press **Control-V**; then type the key you want to assign as the printkey, then an equal sign (=), and finally the ASCII value to be substituted for the printkey during printing. As a result, whenever *SpeedScript 80* encounters the printkey embedded in text, it prints the character with the ASCII value you previously defined. (If you have trouble passing some printkeys to the printer, try adding 128 to the ASCII value you wish to send. Otherwise, some printer interfaces intercept the printkey.)

For example, to define the + key as the letter *z*, first look up the ASCII value of the letter *z* (in either your printer manual or in the BASIC manual). The ASCII value of *z* is 122, so the definition is:

```
+=122<
```

Now, anywhere you want to print the letter *z*, substitute the printkey:

```
Gad+ooks! The +oo is +any!<
```

This will appear on paper as:

```
Gadzooks! The zoo is zany!
```

More practically, here's how you could program italics on an Epson MX-80–compatible printer. You switch on italics by sending an ESC (a character with an ASCII value of 27); then the character 4. You turn off italics by sending ESC 5. So, define * as the escape code (27). Anywhere you want to print a word in italics, bracket it with *4 and *5.

You can similarly define whatever codes your printer uses for features like double width or emphasized mode. For your convenience, four of the printkeys are predefined, though you can change them. The keys **1–4** are defined as character codes 27, 14, 15, and 18, common values for most printers. On most printers, CHR$(27) is the ESC key, CHR$(14) starts double-width printing, CHR$(15) either stops double-width or starts condensed characters, and CHR$(18) usually cancels condensed characters.

Keep one thing in mind about printkeys: *SpeedScript 80* always assumes it is printing to a rather dumb, featureless printer, the least common denominator. *SpeedScript 80* doesn't understand the intent of a printkey; it just sends out its value. So if you make one word within a line double width, it may make the line overflow the specified right margin. There's no way for *SpeedScript 80* to include built-in font and type style codes without being customized for a particular printer, since no set of codes is universal to all printers.

Hints And Tips

It may take you a while to fully master *SpeedScript 80,* but as you do, you'll discover many ways to use the editing and formatting commands. For example, there is a simple way to simulate tab stops, say, for a columnar table. Just type a period at every tab stop position. Erase the line with Control-E; then restore it with Control-^ many times. When you're filling in the table, just use word-left/word-right to jump quickly between the periods. Or you can use the programmable printkeys to embed your printer's own commands for setting and jumping to tab stops.

You don't have to change or define printer commands every time you write. Just save these definitions and load this file for each session. You can create many custom definition files and have them ready to use on disk. You can create customized "fill-in-the-blank" letters. Just type the letter, and everywhere you'll need to insert something, substitute a unique character, such as an * or a Control character. When you're ready to customize the letter, use Find to locate each symbol and insert the specific information. Instead of typing an oft-used word or phrase, substitute a unique character; then use Control-Y to automatically change these characters into the actual word or phrase. You can even use *SpeedScript 80* as a simple filing program. Just type in all your data, flagging each field with a unique character. You can use Find to quickly locate any field.

Using ProDOS *SpeedScript 80*

The ProDOS version of *SpeedScript 80* varies only slightly in operation from the original DOS 3.3 version. Because ProDOS is stored in the

upper 16K of memory, and because BASIC.SYSTEM is not needed for *SpeedScript 80*, you have about 10K more of text memory in ProDOS than you do with DOS 3.3.

All changes in the instructions for using the ProDOS version of *SpeedScript 80* are to support ProDOS filename conventions. Control-C still displays a disk catalog. But the filename convention for Load, Save, and the G (goto file) printkey command has changed. If you boot *SpeedScript 80* off its own startup disk, it remembers the volume name of the disk it was booted from. You don't have to enter a volume name to load, save, or link to a file on that disk. Just enter any legal ProDOS filename. Remember that—unlike DOS 3.3—ProDOS does not allow spaces in filenames. Using spaces in a filename courts disaster.

To access a file on any other disk, though, you'll have to enter the volume name. For example, if you want to load the file named SPROITZ from a disk with the volume name BLANK27, enter /BLANK27/SPROITZ at the LOAD: prompt. Just surround the volume name with slashes. If you have two drives, ProDOS automatically searches for the indicated volume name on both drives. ©

Printer Master

David Stanton

While the dot matrix printer is a wonderfully flexible device, accessing its many features can sometimes be confusing. This convenient, menu-driven program helps you get the most out of your printer by simplifying the process of choosing special printer features. For all Apple II series computers with a dot matrix printer. Disk File: PRINTER.MASTER.

Buying a printer has always been a bit complicated. Should it have a serial or a parallel interface? What printers will be compatible with your brand of computer? What is a reasonable price? Which printers offer the best value for the money? Should you select a dot matrix machine for its speed and its graphics capability, or opt for the letter-quality print of the daisy wheel?

Many magazines have run articles comparing printers of all types, feature for feature. After some investigation, most of us end up with a printer that suits our needs. For many, that means a medium-priced dot matrix printer.

Except for the most demanding purchaser, the near-letter-quality font of modern dot matrix machines can even prove acceptable for occasional business use. As a bonus, the programmability of these inexpensive workhorses provides a flexibility unmatched by the best daisy wheel printers. Unfortunately, a new purchaser's elation is too often dampened after discovering the pages in the user's manual that explain all the control characters required to implement those fancy fonts, foreign alphabets, and optional character sets.

Enter "Printer Master." This menu-driven program remembers those optional features along with their confusing control characters and allows you to configure your printer exactly the way you want it without the necessity of understanding all the details outlined in your printer's manual.

While Printer Master does not access some of the more obscure printer functions, it does allow easy access to foreign alphabets and special character set combinations. Simply run Printer Master, selecting your desired font. Then use your word processor to print old or new text files. With a bit of experimentation, you should be able to achieve exactly the desired results for every occasion.

Since Printer Master is menu-driven and completely self-prompting, you won't need elaborate instructions to use it. Just follow the onscreen prompts and respond to the questions as indicated.

When you run the program it divides the screen into two parts. The upper section displays a series of menus listing the available features. As you select those items, the bottom portion will display the current status of each option. Occasionally the status will not change when an item is selected. This occurs either because the printer cannot use that feature or the status of

Variables

Each of the following variables must be the decimal number that sets the mode indicated. While the values vary, each printer's manual has a chart that shows the proper values for its available features. If your printer does not have a particular feature, simply set that value to 0.

A1	ASCII (U) character set
A2	ASCII (S) character set
BG	block graphics character set (when available)
BR	British character set
CS	prepares character set for new selection
DA	Danish character set
DQ	sets print quality to data (low quality/high speed)
DS	enhanced mode off
DU	Dutch character set
DW	sets double width (elongated)
EL	sets elite mode (12 CPI)
EM	sets emphasized print mode
EN	sets enhanced print mode
EO	turns off emphasized print mode
FC	French/Canadian character set
FI	sets condensed mode (17 CPI)
FR	French character set
G1	sets graphic density to single density graphics
G2	sets graphic density to double density graphics
GA	Greek character set (C. Itoh)
GE	German character set
IO	Italic mode off
IB	IBM character set code (when available)
IL	Italian character set
JA	Japanese language set
L1	sets lines per inch to 6
L2	sets line per inch to 8
LS	Temporarily holds selected language code. This value is used by the language routine. It is not set in the printer parameter lines.
NL	sets printer to near letter quality (Bold, if no NLQ)
NO	establishes standard (normal) character set
NW	Norwegian character set
PC	informs printer that next value is a printer command
PI	sets pica mode (10 CPI)
RE	resets printer defaults
SL	sets italic (slant) mode
SP	Spanish character set
SW	Swedish character set

String Variables

CS$	current character set selection
EM$	Y or N condition of emphasized print
EN$	Y or N condition of enhanced print
GD$	current graphic density selection
LI$	current line spacing selection (6 LPI/8 LPI)
LS$	current language set selection
PQ$	current print quality selection
PR$	number of printer type (not user modifiable)
PS$	current print size selection

a different selection precludes the use of that option. For example, Okidata printers prohibit the use of near-letter-quality print in condensed mode.

To check the appearance of a font selection, choose "Print Test" from the main menu. Printer Master prints a short message to help you diagnose any problems. If anything is not as it should be, the most likely reason is improper variable values; in the next section, we'll learn how to change them.

Making Adjustments

As listed, the program is designed to work with the following popular printers:

Apple Imagewriter
Okidata Microline series
Epson FX series
C. Itoh 8510 series
BMC dot matrix printer

Some models from these manufacturers offer different options. As a rule, Printer Master is compatible with the most versatile printer from each manufacturer. For models that lack some of those advanced features, you may want to replace the appropriate variable values with zeros so you'll have an accurate indication of the printer's capabilities in the configuration display at the bottom of the screen. If you attempt to use a feature that is unavailable on your printer, Printer Master could produce unpredictable results.

For example, the Microline 182 lacks near-letter-quality (NLQ) print mode. By placing a zero in the variable NL, you can assure that the program will not attempt a nonexistent NLQ mode. Don't be afraid to experiment. Turning the printer off and on again always resets it to its original state no matter what commands the program may have sent.

For those who have a different printer, the explanation of the relevant variables should allow you to configure the program to fit your machine. Find the section in your manual that lists the decimal values needed to control each function, and insert those values in the appropriate variables listed in lines 450–460. Table 1 explains the functions of all the relevant variables.

Using this method, you should have little trouble creating a version for any dot matrix printer to which you may have access. Be sure to use proper values, though, or be prepared for strange results. No harm will be done to the printer, but the printout can look very peculiar.

Interfaces, Dip Switches, And Power-Up Defaults

Those who own a printer may already know something about interfaces. Third party interfaces offer a vast array of programmable and switch-selectable configurations. Printer Master does not affect your interface at all.

Printers, too, can be programmed and modified. Internal dip switches control the power-up default conditions that determine how your printer works when you first turn it on. Printer Master assumes that your printer's switches are set in the standard mode established at the factory. If this is not the case, you may need to change the string variable values listed below to conform to your machine's power-up condition. This is neither as technical nor as difficult as it may sound. These variables are defined in line 2140 of the program.

Once you have the minor adjustments necessary, Printer Master will help you show off all the powerful features of your dot matrix printer. Now you can print in a wide range of font sizes and print quality. And foreign language printing becomes simple as well. To print longer and wider spreadsheets, simply choose a smaller print size. With Printer Master you are likely to enjoy using your printer much more. ©

We Won't Be UNDERSOLD

* On items marked *"No One Sells This ----- For Less"*. Copy of valid ad required.

15 Day Free Trial • 90 Day Immediate Replacement Policy • Prices Expire 12-31-87

Complete Apple® Compatible

Computer System

Word Processing For • Home • Business • School

Two Apple® computers in one. Runs virtually all Apple IIc and IIe programs (over 10,000).

640K System Option

Sale $599

Sale

Look At All You Get for only $499.00

Ask about our color monitor and printer upgrade specials.

Apple® is the registered trademark of Apple Computers Inc.

		List Price
1	**Laser 128 Apple Compatible Computer**	$499.00
2	**Big Blue Printer**	199.00
3	**12" 35 MHz Hi-Resolution Monitor**	229.95
4	**Magic Window IIe wordprocessor**	150.00
5	**Magicalc Spreadsheet**	150.00
6	**Magic Memory Data Base**	60.00
7	**Laser 128 Big Blue Printer Interface**	19.95
8	**2 Rolls of Heat Transfer Paper**	19.95
	Comparable Apple System ($2495.00)	$1327.85

Complete System

$499

(Add $35.00 shipping & handling.*)

All 8 Pieces Only $499

Magic Window IIe: Word processing is easy and more efficient because of a simple-to-read menu system and editing commands. Many powerful features of professional quality programs.

Magicalc: Rated as the best electronic spreadsheet on the market for Apple. The speed with which Magicalc solves number problems allows you to accurately analyze decisions.

Magic Memory: File anything, any way you want to: names, addresses, important numbers, dates and notes. Your information is organized simply, and easy to maintain.

Call

(312) 382-5050

We Love Our Customers!

Mail

COMPUTER DIRECT

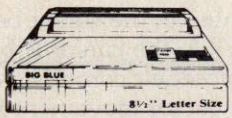
22292 N. Pepper Road
Barrington, IL. 60010

1st In Price, Support, & Warranty

15 Day Free Trial • 90 Day Immediate Replacement Policy • Prices Expire 12-31-87

8½" Letter Size 80 Column Printer

We Liked This Printer So Much,
We Bought Out The Factory!



Sale $39.95

*Add $ 7.50 Shipping.

List $199

- **Big Blue Dot Matrix Printer**
- **Heat Transfer • Enlarged**
- **Graphics (Apple, Atari, & CBM)**
- **Underline • Upper & Lower Case**
- **True Descenders • Adapters For:**
- **Commodore & Atari . . Sale $19.95**
- **Apple IIc & Laser 128 . Sale $12.95**
- **RS-232 Adapter . . . . . Sale $12.95**
  (Please Specify Male or Female)

Pro 20 Daisy Wheel Printer

Wide Carriage Letter Quality
Daisy Wheel Printer

Sale

Sale $99.95

*Add $10.00 Shipping.

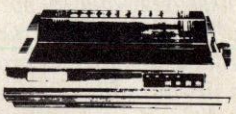
List $499

- **Daisy Wheel Printer**
- **18 CPS Shannon & 22 CPS AAA Text**
- **Print Wheel (Courier 10) & Cartridge Compatible with Diablo® & Qume®**
- **Friction Feed • 13" Form Width**

Extra Daisy Wheels . . . . . Sale $ 9.95
Parallel Centronics Interface (IBM®PC, Apple®IIc, Laser 128, CBM . **Sale $19.95**

10" Comstar 1000 Printer

With Near Letter Quality
Best Quality In The U.S.A.



Sale $139.95

*Add $10.00 Shipping.

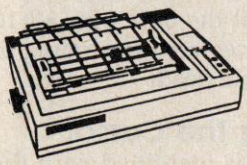
List $349

- **100 CPS Draft & 20 CPS NLQ**
- **Serial Impact Dot Matrix • Bold**
- **Underline • Condensed • Italics**
- **Pica • Elite • Double Strike**
- **Dot Addressable Graphics**
- **Automatic Paper Loading**
- **Tractor & Friction Feed**
- **Superscript & Subscript**
- **Right & Left Margin Settings**

No One Sells These Printers For Less!

NLQ 180 (180 CPS) Printer

Near Letter Quality
Lifetime Warranty*



Sale $189.95

*Add $10.00 Shipping.

List $499

- **Near Letter Quality Selectable From Front Panel • 8K Buffer**
- **High Speed Dot Matrix**
- **Letter Quality Modes • Italics**
- **Elite • Pica • Condensed**
- **Graphics • Tractor & Friction Feed**
- **Lifetime Warranty on Print Head***
- **6 Month Immediate Replacement Policy For Printer**

Hi-Speed LV-2010 (200 CPS) Printer

High Speed Printing (200 CPS Draft)
with Crisp Near Letter Quality

Save

Sale $209.95

*Add $10.00 Shipping.

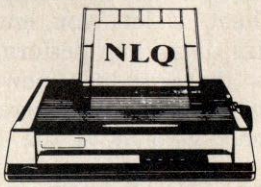
List $499

- **High Speed 200 CPS Draft**
- **IBM® Compatible • Dot Matrix**
- **Near Letter Quality Printing**
- **Standard Pull Tractor & Automatic Single Sheet Loading**
- **Standard Parallel & Serial Interface Ports**
- **Ultra High Resolution Bit Image Graphics • Continous Underline**

300 CPS Printer

300 CPS Draft - 50 CPS NLQ

Comstar 1300 High Speed Printer
With Color Printing Capabilities



Sale $299.95

*Add $10.00 Shipping.

List $599

- **300 CPS Draft - 50 CPS NLQ**
- **Superb Near Letter Quality**
- **Variety Of Characters/Graphics**
- **IBM & Epson Modes • 10K Buffer**
- **Auto Paper Loading & Ejection**
- **Parallel & Serial Interface Ports**
- **Download Character Setting**
- **Front Panel Margin Setting**
- **Optional 7-Color Printing Kit...$99.95**

COMPUTER DIRECT (A Division of **PROTECTO**)
22292 N. Pepper Road, Barrington, IL. 60010
Call (312) 382-5050 To Order
We Love Our Customers!

* Illinois residents add 6½% sales tax. All orders must be in U.S. Dollars. We ship to all points in the U.S., CANADA, PUERTO RICO & APO-FPO. Please call for charges outside continental U.S. or C.O.D. **MAIL ORDERS** enclose cashier check, money order or personal check. Allow 14 days delivery, 2 to 7 for phone orders and 1 day express mail. Prices and availability subject to change without notice. (Monitors only shipped in continental U.S.)

VISA — MASTERCARD — C.O.D.

SpeedCalc
For Apple II Computers

Kevin Martin

This professional-quality spreadsheet has all the important features you'd expect from a commercial program. In addition, its data files can be merged into text files created with the Apple SpeedScript *word processor. Apple* SpeedCalc *requires a disk drive, and a printer is optional but recommended. Disk File: SPEEDCALC, SPEEDCALC3.3*

Have you ever planned a budget for your home or office? If so, you probably used some sort of worksheet divided into rows and columns. Perhaps you wrote the months of the year along the top of the sheet and listed categories for earnings and expenses along one side. After entering data for each category and month of the year, you could calculate total income figures by adding or subtracting numbers in each of the sheet's "cells."

That's a classic example of a worksheet. It lets you enter and organize data, then perform calculations that produce new information. A *spreadsheet* program is an electronic version of the familiar paper worksheet. Since it does all the calculations for you at lightning speed, an electronic spreadsheet is far more convenient than its paper counterpart. And spreadsheet programs also offer editing features that let you enter and manipulate large amounts of data with a minimum of effort.

Apple *SpeedCalc* is an all machine language spreadsheet program for Apple II computers with either DOS 3.3 or ProDOS. Though relatively compact in size, *SpeedCalc* is fast, easy to use, and has many of the features found in commercial spreadsheet programs. Even better, the "*SpeedScript* File Convertor" program lets you merge your *SpeedCalc* files into word processing documents created with *SpeedScript*.

Working together, *SpeedCalc* and *SpeedScript* make a powerful team. You can merge a chart of sales figures into a company report, create a table of scientific data for a term paper, and manipulate numeric information in many other ways. In a sense, a spreadsheet program brings to arithmetic all of the flexibility and power that a word processor brings to writing.

To start the DOS 3.3 version, first enter **BLOAD SPEEDCALC3.3** (replace SPEEDCALC with the appropriate filename if you used some other name when saving the program). After the program loads, simply type RUN as you would for a BASIC program. To start the ProDOS version of *SpeedCalc*, first boot ProDOS, then enter **−SPEEDCALC**. This removes the BASIC interpreter and lets *SpeedCalc* take over the system.

If you're using an Apple IIe or IIc, be sure the Caps Lock key is down: *SpeedCalc* doesn't accept lowercase text input.

The Apple *SpeedCalc* Screen

SpeedCalc uses the top line of the screen as the *command line*. This is where *SpeedCalc* displays messages and asks you questions.

Screen lines 2–4 are the *input buffer* area. This is the work area where you enter and edit data. As you'll see in a moment, the input buffer also displays the data contained in the current cell. The work area cursor is an inverse less-than symbol (<). When the cursor is solid (nonblinking), *SpeedCalc* is waiting for a command or for data to be entered. After a character of data has been entered, the cursor begins blinking. While the cursor is blinking, most *SpeedCalc* commands (except for the cursor movement keys) are deactivated until you press RETURN to enter the data into the worksheet.

The lower 20 screen lines are your window into the spreadsheet. Though the spreadsheet contains many rows and columns, only a few can fit on the screen at one time. By scrolling the screen back and forth with the cursor, you can move the display window to any part of the spreadsheet.

The *SpeedCalc* worksheet consists of 50 vertical columns labeled with letters (AA, AB ... BX) and 200 horizontal rows numbered from 1–200. The rectangle where a row and column intersect is called a *cell*. Cells are where you store data. With 50 columns and 200 rows, the *SpeedCalc* spreadsheet has a maximum of 10,000 (50*200) cells. Due to memory limitations, however, only about a third of these can actually contain data. But you may spread out the data over all 10,000 cells if necessary, depending on the format you need.

Moving The Cursor

Each cell is identified with the letters of its column and the number of its row. For example, the cell at the extreme upper-left corner of the sheet is called AA1, since it's in column AA and row 1. The cell below that is AA2. Moving one cell to the right from AA2 puts you in cell AB2, and so on.

Your current position in the spreadsheet is shown by the highlighted cursor. The simplest way to move around the sheet is with the cursor keys (on the Apple II or II+, use CTRL-K to move up and CTRL-J to move down). Another way to move the cursor is with CTRL-@

(CTRL-SHIFT-P for the Apple II or II+, or CTRL-2 for the Apple IIe and IIc.) Press CTRL-@ once to "home" the cursor on the current screen: The cursor moves to the upper-left cell. Press CTRL-@ twice in succession to move the cursor to cell AA1, the home position for the entire sheet.

SpeedCalc also has a *goto* command for moving the cursor over long distances. When you press CTRL-G, the command line displays GOTO: followed by an underline cursor. The underline cursor generally indicates that *SpeedCalc* is waiting for data—in this case it expects the name of the cell where you wish to go. If you enter **BA188** at this point, *SpeedCalc* moves the cursor to the cell at column BA in row 188, adjusting the screen window as needed. Take a few moments to practice moving around the spreadsheet with all three methods; you'll be using them a lot. In a later section, we'll discuss how to change the size and format of a cell.

Keyboard Commands

SpeedCalc offers many different commands, a few of which are entered by pressing one key. However, most commands are entered by pressing CTRL along with another key. CTRL-G, as you've seen, is the goto command. CTRL-A displays the amount of free memory available, and so on. The most drastic command is CTRL-X, which exits *SpeedCalc* and reboots the system. Since this effectively erases all data in memory, *SpeedCalc* prompts you with ARE YOU SURE Y/N? before it shuts down. To cancel the command, simply type N (or any key other than Y).

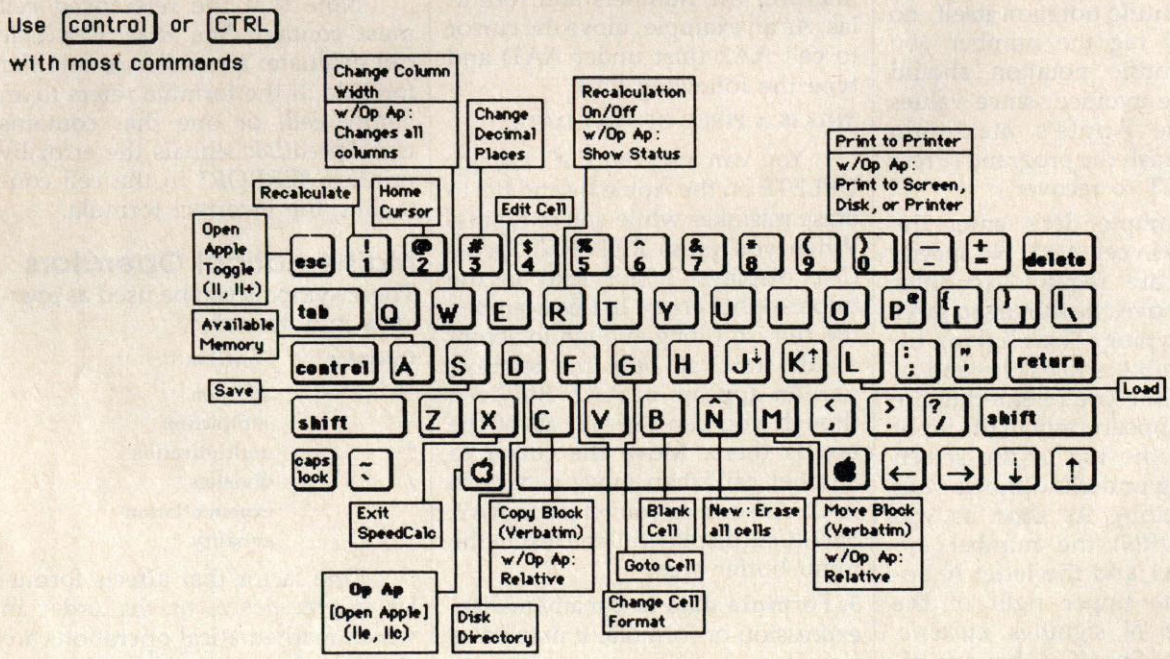
A few commands require you to press three keys at once. This sounds more awkward than it is in practice, since two of the three keys are Open Apple and CTRL. For instance, the *relative copy* command is performed by pressing Open Apple-CTRL-C (hold down Open Apple and CTRL, then press C).

The older Apple II and II+ models don't have an Open Apple key, so ESC is programmed to act as an Open Apple toggle. Pressing ESC once makes all following keypresses behave as if they were preceded by Open Apple. Pressing ESC again turns off this effect. In this article, wherever the instructions call for the Open Apple key, Apple II and II+ owners should instead precede the keypress with ESC, then use ESC again afterwards to disable the Open Apple toggle. For example, the command to check the recalculation status is Open Apple-CTRL-R; Apple II and II+ owners should instead press (and release) ESC, then press CTRL-R. There's no visible indication that the Open Apple toggle is in effect, so you must use ESC carefully or your keypresses will have unexpected results. For safety, always remember to press ESC again to toggle this function off after using a command that requires Open Apple. The table lists all the *SpeedCalc* commands, and the figure shows the keyboard layout with a description of what each key does. We'll be discussing each command in more detail below.

Three Data Types

Before entering any data, you must know what kind of data *SpeedCalc* accepts. There are three different types: numbers, text, and formulas. Let's look at each type in turn.

1. Numeric data consists of numbers—the basic stuff that spreadsheets work with. *SpeedCalc* has a few simple rules for numeric data: A number must be a decimal value (base 10, not hexadecimal) composed of one or more digits from 0–9, with an optional plus or minus sign. A decimal point is also optional. If you include any other characters in numeric input, *SpeedCalc*



treats the entire input as text data (as explained below). Thus, the numbers 123, .001, and −65535 are valid numeric data. The number 65,535 is invalid because it includes a comma.

The allowable range for numbers in Apple *SpeedCalc* is the same as for Applesoft, roughly −1.7E38 to +1.7E38. If a calculation produces a number outside the allowable range, you'll see the message *ERROR* in the cell containing the formula. This doesn't happen very often, since *SpeedCalc* won't let you enter a number more than 36 digits long, and there's rarely a need to use such large numbers unless you're tracking the national debt.

Although an input value can be up to 36 digits long, numbers in *SpeedCalc* calculations are accurate only to nine digits. This must be taken into account when doing any calculation involving large values. For example, you can enter the value 1122334455.66 into a cell, and the cell holds the value with no rounding. However, if you use the value from that cell in a formula, the value is rounded to nine digits—1122334460.00—and the result of the calculation is accurate only for the first nine digits.

You can enter values in scientific notation by following a number with the letter E and the appropriate power of 10. For example, you can enter 1,234,000 as 1.234E06. However, *SpeedCalc* never uses scientific notation itself, no matter how big the number you enter. Scientific notation should generally be avoided, since values outside the Apple's maximum range will crash the program. (Press CTRL-RESET to recover.)

For example, let's enter the number 123 in cell AA1. No special commands are required to enter data: Just move the cursor to AA1 and begin typing. The blinking inverse < symbol shows the end of the data. While you're entering the number, it appears only in the input buffer near the top of the screen (the blinking underline shows your cursor position). As soon as you press RETURN, the number appears in AA1 and the letter N appears at the upper right of the screen. The N signifies *numeric*, meaning that *SpeedCalc* has accepted the entry as valid numeric data. Move the cursor to a vacant cell, then move it back to AA1. The input buffer displays whatever data is found in the cell under the cursor. When the current cell is empty, the buffer is empty as well.

If you want to change anything during data entry, press the ESC key. ESC always deletes the character before the cursor (or has no effect if the cell is empty). Later on, we'll explain how to edit existing data. Use ESC carefully; remember that when you're not editing (when the cursor is not blinking), ESC acts as an Open Apple toggle. On the Apple IIe and IIc, you can (and should) use DELETE instead of ESC.

As you've seen, pressing RETURN enters a data item into the current cell. You can also end the input by pressing a cursor key. The data is entered as if you had pressed RETURN, and the cursor moves in the indicated direction. This feature is handy for entering a lot of data: Simply type the entry, move the cursor to the next cell, enter more data, and so on.

2. Text data is not "data" in the strict sense, since *SpeedCalc* doesn't use it in calculations as it does numbers and formulas. Text data is there only to help people understand what the other data means. Text may consist of comments, titles, column headings, subheadings, or whatever you need to interpret the numbers and formulas. As an example, move the cursor to cell AA2 (just under AA1) and type the following line.

THIS IS A PIECE OF TEXT DATA.

You can use the ESC key (or DELETE on the Apple IIe and IIc) to erase mistakes while you're typing. When you press RETURN, *SpeedCalc* displays T (for text) in the upper-right corner. In this example, the cell isn't long enough to accept all the text, so only the leftmost portion appears in AA2. But even though you can't see it, all of the text is there. Move the cursor to another cell, then move it back to AA2. As soon as you return to AA2, *SpeedCalc* displays all the text in the input buffer area.

3. Formula data is a mathematical expression or formula. It may be as simple as 2 + 2 or as complex as your imagination (and mathematical prowess) allows. The first character in a formula must always be an equal sign (=). If you omit this symbol, *SpeedCalc* either signals an error or treats the data as text.

The true power of a spreadsheet is that a formula in one cell can refer to another cell. This is easier to demonstrate than to explain. Move the cursor to cell AA3 and type the following line:

=AA1*25.01+@SQR(4)

As soon as you press RETURN, *SpeedCalc* displays F (for formula) in the upper-right corner and puts the *result* of the formula (not the formula itself) in AA3. If AA1 contains 123, the value 3078.23 appears in AA3. In plain English, this formula means "multiply the contents of cell AA1 by 25.01 and add the square root of 4." Before we examine the formula more closely, here's a quick demonstration of what makes a spreadsheet such a powerful tool. Move the cursor back to AA1 and press CTRL-R. The command line displays the message RECALCULATION IS ON, meaning *SpeedCalc* now automatically recalculates the entire sheet whenever you make a change. Now change the number in AA1 to 456 (simply move to the cell and start typing). The new result (11406.56) automatically appears in cell AA3. We'll explain more about automatic recalculation later.

Note that the referenced cell must contain data that *SpeedCalc* can evaluate: a number or another formula. If the formula refers to an empty cell, or one that contains text, *SpeedCalc* signals the error by printing *ERROR* in the cell containing the incorrect formula.

Mathematical Operators

These symbols can be used as *operators* in a formula:

Operator	Function
+	addition
−	subtraction
*	multiplication
/	division
^	exponentiation
=	equality

One factor that affects formulas is *precedence*, or the order in which mathematical operations are performed. In *SpeedCalc*, formula

operators have the same precedence as in ordinary math.

The first operators to be evaluated—those with the highest precedence—are those enclosed in parentheses. Where one set of parentheses encloses another, the expression in the innermost set is evaluated first. The next operators to be evaluated are exponents. Multiplication and division have equal precedence; both operations are lower than exponentiation. Addition and subtraction have the lowest precedence of all. To take one example, *SpeedCalc* evaluates the formula **=5*(8+3*−2)^2−10/+2** as the value 15, just as in ordinary math. Note how the result is affected by the plus and minus signs before the two 2's.

Functions

Formulas may also include any of the functions listed here:

@ABS( )	**absolute value**
@ATN( )	**arctangent**
@AVE( )	**average of a block of cells**
@COS( )	**cosine**
@EXP( )	**natural exponent**
@INT( )	**integer**
@LOG( )	**natural logarithm**
@SGN( )	**sign**
@SIN( )	**sine**
@SQR( )	**square root**
@SUM( )	**sum of a block of cells**
@TAN( )	**tangent**
PI	**value of pi (3.14159265)**

All the functions except PI begin with the @ symbol and are followed by parentheses. The parentheses of a function may contain a number or formula. For example, the formula =@SQR(4) generates the square root of 4. The formula =@SQR(AA1) returns the square root of whatever value cell AA1 contains. Note that the argument (value within parentheses) of the functions @TAN( ), @SIN( ) and @COS( ) must be expressed in radians; the result of the function @ARC( ) is expressed in radians. The function @INT( ) generates an integer (whole number) by truncating (discarding the fractional part of) a numeric value; note that this is different from rounding.

The function @AVE( ) calculates the mean average of the values in a block (group) of cells. The function @SUM( ) calculates the sum of a block. Both functions require you to define the block so *SpeedCalc* knows which cells to include in the calculation. This is done by putting two cell names separated by a colon in the parentheses. The first cell name defines the upper-left corner of the block, and the second defines the bottom-right corner. For instance, @AVE (AA1:AD20) calculates the average of all the cells from AA1 to AD20. The function @SUM (AA1:AD20) calculates the sum of AA1 through AD20, and so on. An error results if any cell in the block is blank or contains text data.

Editing The Sheet

Editing is a very important spreadsheet function. The simplest way to change what a cell contains is to move to it and start typing. The old data in that cell is replaced by whatever you enter. For instance, to replace the contents of cell AA1 with the number 456, move to that cell, type 456, and press RETURN or exit with a cursor key. Press CTRL-B (think of *blank*) to erase what's in the current cell. To erase everything in the sheet, press CTRL-N (think of *new*). Before carrying out this drastic operation, *SpeedCalc* asks you to confirm it by pressing Y or N.

In some cases, only a minor change is needed. *Edit mode* lets you change the data in a cell without retyping the entire entry. To activate edit mode, move to the desired cell and press CTRL-E. In this mode, up and down cursor movement is disabled, and the left/right cursor keys move within the input buffer. Erase unwanted characters with the ESC key (or the DELETE key on the Apple IIe and IIc). Typing in edit mode inserts new characters in the line: Everything to the right of the new character moves right one space (unless the buffer is already full). Since the cursor keys have a different function in edit mode, you cannot use them to end the input. Press RETURN to enter the new data and escape from edit mode.

SpeedCalc displays *ERROR* in a cell when you enter an erroneous formula. Usually this means you've made a typing error in that cell, or the formula refers to text or an empty cell. A line of asterisks (*********) signals that a number is too large to be printed in the cell. Though these messages appear in the cell area, no data is lost. You may move to the affected cell, view its contents in the input buffer, and make whatever correction is needed.

Recalculation

This feature is the very core of a spreadsheet. As you know, entering or editing a piece of data makes *SpeedCalc* perform a calculation and put the result in the cell under the cursor. In most cases, the new data relates to data in other cells, so you'll ultimately want to recalculate the entire spreadsheet as well. This can be done manually or automatically.

To recalculate the spreadsheet manually, enter an exclamation point (SHIFT-1). *SpeedCalc* begins at AA1 and recalculates every cell that contains data, placing fresh results wherever needed. If you switch to automatic recalculation mode, *SpeedCalc* automatically recalculates the entire spreadsheet each time you enter new data or edit what exists. When you press CTRL-R, *SpeedCalc* changes the recalculation status and displays it at the top of the screen. If automatic recalculation was turned off before, it is now on (and vice versa). If you aren't sure which mode you're in, press Open Apple-CTRL-R; *SpeedCalc* displays the mode without changing it.

Automatic recalculation can be fun to watch in a large spreadsheet: Every time you make a change, new results appear everywhere on the screen. However, the more data your spreadsheet contains, the longer it takes to update the entire sheet. For this reason, you may want to turn off automatic recalculation most of the time, recalculating manually whenever you need to view results.

One problem with recalculation arises from the order in which cells are calculated. Because only one cell can be calculated at a time, you must sometimes recalculate the entire spreadsheet two or three times to get correct results in every cell (this is common to all spreadsheet programs). For instance, say you have a formula in AA1 which refers to a formula in AB15. When *SpeedCalc* calculates AA1, it must use the existing data from AB15—which is probably out of date, since the formula in AB15 hasn't been

recalculated yet. To avoid this problem, you should always recalculate a sheet manually two or three times before printing or saving it to disk.

SpeedCalc offers a number of other features. Before experimenting with them, you should spend some time typing in a hypothetical spreadsheet—perhaps a fictitious yearly budget—to become thoroughly familiar with the basic commands covered so far. Most importantly, create formulas using all the operators in different combinations. Try doing things that you know will cause errors. Then correct the errors in edit mode, and so on. It takes a thorough grasp of the fundamentals to get the most out of *SpeedCalc*'s advanced features.

Change Format

The default (normal) format for numeric data is flush right with rounding to two decimal places. In other words, the number is displayed in the rightmost part of the cell, with two numbers after the decimal point. Text and formulas are also displayed flush right. *SpeedCalc* offers several commands for changing cell formats. (Apple II and II+ owners who are using the ESC toggle in place of the Open Apple key should be careful that ESC is not in effect when it's not desired; accidental global changes may be difficult to reverse.)

Change Format (CTRL-F). This command changes the location of data in the cell. When you press CTRL-F, the *SpeedCalc* command line displays the question FORMAT: LEFT, CENTER, OR RIGHT JUSTIFY?. Press L, C, or R to move the data to the left, center, or right of the cell.

Change Decimal Places (#). *SpeedCalc* also lets you change the number of decimal places for any cell. The default number of decimal places is 2, but you may change it to anything from 0–15. Press # (SHIFT-3) to change this value: *SpeedCalc* prompts you to enter a number from 0–15. If you choose zero decimal places, any number in that cell is rounded off to the nearest integer (whole number). If you choose 15, a number in that cell is not rounded off at all—*SpeedCalc* displays it exactly as you entered it or as it was calculated from a formula.

Width (CTRL-W). The width command changes the width of an entire column of cells. Move the cursor to any cell in the desired column, then press CTRL-W. When *SpeedCalc* displays the prompt WIDTH:, respond with a number from 4–36. The entire screen is redrawn to accommodate the new format, and may look very different depending on what value you chose. For instance, if you increase a column's width, the rightmost column of the former display may disappear: *SpeedCalc* only displays as many complete columns as it can fit on the screen. If you decrease the width of a column, you may see asterisks where numbers used to be (indicating the cell is now too small to display the entire number). To get rid of the asterisks, expand the column as necessary.

Global Format (Open Apple-CTRL-F). This is the same as the ordinary format command, but operates globally, changing every cell in the sheet instead of just one.

Global Width (Open Apple-CTRL-W). This is a global version of the width command. Every column in the sheet changes to the designated width.

Macro Editing

After typing in a large spreadsheet, you may decide to make a major change. You may want to add new data somewhere in the middle, delete a section, or move a group of cells from one location to another. *SpeedCalc*'s macro (large-scale) editing commands simplify such operations, affecting an entire block of cells at once. A *block* is simply a group of cells connected in rectangular fashion. You can define it as a single cell, a row or column, or any rectangular area within the spreadsheet.

There are two ways macro commands work: *verbatim* or *relative*. To take a simple example, say that cell AA2 contains the formula =AA1*5 and you want to move its contents to cell AB2. When this is done in verbatim mode, AB2 contains an exact copy of what was in AA2 (=AA1*5). Note that the cell name used in the formula does not change: The formula still refers to AA1. If you perform the same operation in relative mode, the cell name in the formula is adjusted to fit the new location. In this case, AB2 would contain the formula =AB1*5. (Apple II and II+ owners who are using the ESC toggle in place of the Open Apple key should be careful that the toggle is not in effect when not desired; accidental relative changes can lead to problems that are difficult to detect and correct.)

Copy (CTRL-C). The copy command copies a block of cells into a different location without disturbing the original cells. Place the cursor on the upper-left corner of the block you want to copy, then press CTRL-C. *SpeedCalc* prompts you to move the cursor to the lower-right corner of the block you want to copy. Once the cursor is in place, press RETURN. Now *SpeedCalc* prompts you to move the cursor to the place where you want to put the block: This is the upper-left corner of the new position. Once the cursor is there, press RETURN again. The new data replaces whatever was contained in the designated cells. Note that if you define an impossible block (for instance, moving the cursor to the upper-left of the original position, rather than below and to the right), *SpeedCalc* does not copy any data. This provides a way to cancel the command if you press CTRL-C accidentally.

Move (CTRL-M). This command works like a copy, but it fills the original cells with blanks. Though *SpeedCalc* has no express insert command, you can use this command to make space for new data in the middle of a spreadsheet. Simply move everything below the insertion point down as far as you need.

Because RETURN generates the same character code as CTRL-M, you may find when you first begin using *SpeedCalc* that you accidentally invoke the move function by pressing RETURN when you shouldn't have. To cancel this, simply press RETURN twice more without moving the cursor.

Relative Copy (Open Apple-CTRL-C). This form of the copy command adjusts the cell names used in formulas within the copied block (see explanation above).

Relative Move (Open Apple-CTRL-M). This is the relative form of the move command. Cell names in formulas are adjusted to reflect the move.

Memory Management

The DOS 3.3 version of *SpeedCalc* makes about 12K (over 12,000 characters) of memory available for data; the ProDOS version provides approximately 17K. As noted earlier, *SpeedCalc* lets you spread your data out over a much larger number of cells than you can actually fill with data. The extra space is provided to give you full control over the final format of the spreadsheet and to leave some elbow room for move and copy operations.

Because memory is limited, you should keep careful track of how much is free while using the program. Press CTRL-A to display the amount of free memory. We suggest limiting your spreadsheets to 1,600 cells (equivalent to 40 rows by 40 columns) when using the DOS 3.3 version, or 2,500 cells (a 50 × 50 worksheet) when using the ProDOS version. If you've filled nearly all of free memory, you may have to break the spreadsheet into two smaller sheets.

Although *SpeedCalc* checks the amount of available memory and displays an error message if you run out, you should be careful not to exhaust free memory. Any move or copy operation in process will be aborted if sufficient memory is not available.

Disk Operations

SpeedCalc has three disk commands which allow you to save a spreadsheet to disk, load it, and display the disk directory. The directory command is the simplest to use: Simply press CTRL-D. The spreadsheet disappears and a directory of the disk in drive 1 is displayed. Press RETURN to return to the spreadsheet.

To save a spreadsheet to disk, press CTRL-S. *SpeedCalc* prints SAVE: on the command line, followed by an underline cursor. Enter a valid Apple filename and press RETURN. (If you change your mind and decide not to save anything, press RETURN without typing a filename.) If no disk error occurs while the spreadsheet is being saved, *SpeedCalc* displays NO ERRORS in the command line and returns you to command mode. If there was an error, you'll hear a beep and see the message I/O ERROR in the command line.

To load a saved file from disk, press CTRL-L. Again, you can cancel the operation by pressing RETURN without entering a filename. *SpeedCalc* prompts you to enter the filename and displays the error status when the operation is complete.

When saving or loading *SpeedCalc* files with ProDOS, you must specify the prefix along with the name. If you don't want to type the prefix every time you enter a filename, simply call up a directory for the disk you want to use to save or load. This automatically sets the prefix to match the current disk, relieving you of the need to enter it with every name.

Printing

SpeedCalc lets you print data to three different devices: to the screen for previewing output, to a printer for permanent documentation, or to a disk file for integrating the data with a *SpeedScript* document.

To print a hardcopy of the spreadsheet to a printer in slot number 1, press CTRL-P. *Before using this command, you must position the cursor below and to the right of the block of cells you wish to print.* The upper-left corner of the printout starts at cell AA1.

To send output to a printer with a slot number other than 1, or to the screen or a disk, first position the cursor in the lower-right corner of the block you want to print. Then press Open Apple-CTRL-P (toggle ESC on the Apple II and II+). *SpeedCalc* asks if you want to print to the screen, to disk, or to the printer. Press S to preview output on the screen, D to print to disk, or P to select printer output. Pressing any other key cancels the command.

If you select the P option after pressing Open Apple-CTRL-P, *SpeedCalc* asks you specify a slot number by pressing one of the number keys from 1–7. This permits you to use a printer in any of those slots. If you change your mind at any point during this process, press RETURN without entering anything; *SpeedCalc* returns you to command mode.

You can also print *SpeedCalc* data to a disk file for use in a *SpeedScript* document. Select the D option after pressing Open Apple-CTRL-P, then enter a filename. The data is saved as a disk file of that

SpeedCalc Commands

Command	Action
CTRL-A	available memory check
CTRL-B	blank (erase) current cell
CTRL-C	copy block verbatim
CTRL-D	disk directory
CTRL-E	edit current cell
CTRL-F	change cell format
CTRL-G	goto selected cell
CTRL-L	load *SpeedCalc file*
CTRL-M	move block verbatim
CTRL-N	new (erase entire sheet)
CTRL-P	print file on printer
CTRL-R	turn recalculation on/off
CTRL-S	save *SpeedCalc* file
CTRL-W	change column width
CTRL-X	exit *SpeedCalc*
CTRL-@	home cursor
Open Apple-CTRL-C	copy block relative
Open Apple-CTRL-M	move block relative
Open Apple-CTRL-P	print to screen, disk, or printer
Open Apple-CTRL-R	check recalculation status
Open Apple-CTRL-W	change width of all columns
! (SHIFT-1)	recalculate sheet
# (SHIFT-3)	change decimal places

Note: The Apple II and II+ have no Open Apple key, so ESC must be used as an Open Apple toggle. Pressing ESC once makes all following keypresses behave as if Open Apple were pressed. Press ESC again to turn off the Open Apple toggle.

name. Note that *printing* to disk creates a different type of file than *saving* to disk, and *SpeedCalc* cannot reload files in the print format. You should *save* files you wish to reload into *SpeedCalc*, and *print* files you wish to convert for *SpeedScript*. Unlike the *SpeedCalc* save and load commands, no error messages are provided if the spreadsheet cannot be printed to disk. Thus, you must ensure that the drive contains a write-enabled disk with sufficient space to hold the printed spreadsheet before you attempt to print to disk.

SpeedScript File Converter

SpeedCalc sends data to the printer in simple, plain vanilla form. That may be fine for personal use, but if you're creating a document for others to view, you may want special features such as boldface, underlining, etc. Since Apple *SpeedScript* offers a way to access these features (and many more), no attempt has been made to include them in *SpeedCalc*. All that's needed is a simple program to convert *SpeedCalc* files into a form that *SpeedScript* can load. Then you can edit the file with *SpeedScript* as you would any other document—inserting printer control codes, reformatting the text, merging it with other text, and so on. Use the *SpeedScript* file converter program to perform the conversion. Here are the steps to follow to convert a *SpeedCalc* file for *SpeedScript:*

1. After creating a spreadsheet with *SpeedCalc*, print it to disk as described above.

2. Exit *SpeedCalc*, then load and run *SpeedScript* File Converter. The program prompts you to enter the name of the *SpeedCalc* file you printed to disk. Then it asks you to enter the name of the *SpeedScript* file you want to create (of course, this name should be different from the first). The File Converter then constructs a *SpeedScript*-loadable disk file from the *SpeedCalc* file.

3. After the File Converter is finished, load and BRUN *SpeedScript*, then load the new *SpeedScript* file as you would any *SpeedScript* document. The data appears on the screen, ready to be edited in any way you wish. ©

Skyscape

Robert M. Simons
Version by Tim Victor

This unique program, written by a planetarium director, presents the sky as it can be viewed at any date and time from the year 1977 onward—including zodiac constellations and all the visible planets. It also calculates planet tables, positions of the sun, and phases of the moon for any date and time from 1977 into the future. Disk File: SKYSCAPE.

For thousands of years the sun, moon, and planets in our solar system have excited human imagination. In ancient times they were regarded as gods whose distant motions influenced the course of earthly events. Though we now understand more about the true nature of celestial objects, many facts remain unknown, and a brilliant nighttime sky still presents an inspiring spectacle.

Whether you're seriously interested in the sky or just casually curious, "Skyscape" is a convenient tool for extending your knowledge. It opens a movable window on the heavens, displaying the position of our sun, moon, and neighboring planets from almost any location on Earth, at any point in time from 1977 into the distant future. Since it performs all the necessary calculations, you can enjoy and learn from this program even if you're not an expert in astronomy. In addition to providing data about the position of celestial objects, it draws a sky map on the screen, showing each object as it would appear to you at the chosen location and time.

To get started, type in the appropriate version of Skyscape for your computer and save a copy before running it.

Past, Present, Or Future

Skyscape begins by asking you to answer several questions. Enter the year, choosing any year from 1977 forward. In some ways this is the most important input of all, since objects in our solar system move significantly from one year to the next. After you choose the year, Skyscape allows you to enter the month and day.

Next you must enter the latitude (north/south position on Earth) from which you wish to view the sky. Latitude 0 places you, the observer, at the equator. Latitudes 1–90 place you in the northern hemisphere (north of the equator). To choose a southern latitude (south of the equator), enter a negative number from −1 to −90. Skyscape generally represents southerly locations with negative values.

Whenever Skyscape asks for information, it checks your entry to make sure it's in the acceptable range. If you enter an illegal value, the program displays an error message and gives you another chance.

The Sun And Moon

Though very different in size and composition, the sun and moon are alike in being the largest celestial objects visible from Earth. After you enter the date and latitude, Skyscape displays a table of data for the sun and moon. In addition to the date, day of the year, and

latitude north or south, you'll see the following information:

• Sun's geocentric angle. This figure represents the sun's position as a number of degrees relative to the vernal equinox. The vernal equinox is where the sun is located when spring begins in the northern hemisphere (the same time that autumn begins in the southern hemisphere).

• Sun's declination. The number of degrees north or south of the equator. Negative values indicate a southerly location.

• Sun's altitude at noon. The location of the sun in degrees from the northern or southern horizon at noon.

• Sun's right ascension. Just as longitude and latitude indicate locations on the Earth, *right ascension* and *declination* are used to pinpoint locations in the sky. For this purpose the sky is visualized as a gigantic sphere surrounding the Earth. Declination locates a point vertically in the celestial sphere and right ascension locates it horizontally. Right ascension values are given in *hours* and *minutes* in the range 0:00–23:59. Right ascension 0:00 is exactly at the vernal equinox. Larger right ascension values lie to the east of smaller ones.

• Right ascension at 9 p.m. The right ascension which would be on the meridian at 9 p.m. This coordinate system would be found on star charts. By comparing this number with those charts, you can tell what stars and constellations would be visible at that time.

• Moon's age. The number of days since the last new moon.

• Moon's elongation. The location of the moon in degrees east or west of the sun.

• Moon's phase. The phase of the moon on this particular day.

The Planet Table

After viewing the sun and moon display, press P to continue to the next display screen, which contains the planet table. (Press D if you wish to enter a new date.) The planet table shows vital information about the visible planets (through Uranus, which is at the limit of our visibility). The table shows the position of each planet in right ascension and degrees east or west of the sun. It also shows the distance of each planet from Earth in millions of miles.

If you'd rather see the distance in kilometers, modify the program to change the value of ES=93 to ES=149.6 in line 80.

Some planets have an asterisk to the left of the right ascension figure. This signifies that they are visible at 9 o'clock this evening. For reference, the planet table also includes the sun's present right ascension and its right ascension at 9 p.m. Press D to input a new date or S to view a graphics display of the sky at any time in the current day.

The Visible Skyscape

After selecting the sky display, you must enter the hour when you wish to view the sky. The hour value should be a whole number from 0–23 (enter 22 for 10 p.m., etc.). You'll also need to enter the minutes (0–59). Skyscape then displays the time and offers you a chance to enter different values. Press RETURN or Enter when you're satisfied with the time.

Skyscape now displays the sky as it would appear at the chosen latitude, date, and time. Since the sky looks very different from different places on Earth, the latitude affects the display considerably. If your latitude is in the range 24–90 degrees north or south, the sky shows a dashed line representing the position of the celestial equator, along with symbols representing the sun, moon, and planets visible at that time. If your latitude is in the tropical region—from 23½ degrees north to 23½ degrees south—the dashed line indicates a position directly overhead.

If you're viewing in the northern hemisphere, north is above the dashed line and south is below it. In the southern hemisphere these directions are reversed. Below the sky display is a key that interprets the symbols used to represent celestial objects. If more than one object is positioned at the same spot, the symbols are displayed above each other.

At the bottom of the sky you may see two-letter abbreviations. These represent zodiac constellations that would be visible from your chosen vantage point. Skyscape uses the abbreviations AR (Aries), PI (Pisces), AQ (Aquarius), CP (Capricorn), SA (Sagittarius), SC (Scorpio), LI (Libra), VI (Virgo), LE (Leo), CA (Cancer), GE (Gemini) and TA (Taurus). Each constellation is located above the spot where its abbreviation appears. In northern latitudes, the border of each constellation's zone begins at its abbreviation and extends left. In southern latitudes, the constellation extends right from the position of its abbreviation.

Daytime skies are shown in blue and nighttime skies in black. Skyscape does not calculate the actual rising or setting time of the sun. Average rising and setting times of 6 a.m. and 6 p.m. are used in every case. You may obtain exact rising and setting times from local newspapers. However, keep in mind that there is usually about an hour of twilight before sunrise and after sunset.

Halley's Comet

In addition to permanent objects, Skyscape's graphics display includes Halley's Comet, which was visible during late 1985 and early 1986. If you choose a date from November 1, 1985 to May 29, 1986, Skyscape calculates the position of Halley's Comet and includes it in the graphics display (if it would have been visible at the place and time you select). The comet's position is based on the best predictions available at the time of this writing (summer 1985).

While Skyscape is generally accurate, it bases most position calculations on circular orbits. This introduces a certain element of error, since no object in our solar system has a perfectly circular orbit. The position error is most pronounced for Mercury and Mars (whose orbits are quite elliptical), but does not significantly affect other objects. I've found Skyscape accurate enough for my own purposes, which include planning astronomy classes and planetarium displays.©

Fast Filer

Richard Mansfield, Editorial Director and
Patrick Parrish, Programming Supervisor

Maintain a master index of magazine articles with this short BASIC program. Disk File: FAST.FILER.

How many times have you been working on a program when you recall a magazine article that has just the information you need—but finding it is another matter? That is, you know the article's *somewhere* in the house—but where? You could spend hours paging through back issues to find what you're looking for. Now, with "Fast Filer," you'll have a fast and easy way to retrieve such information.

Searching The Database

Fast Filer is designed for simplicity and convenience. To search the database, all you really need to do is type RUN and follow the prompts. The program first asks whether you want to send output to the screen or the printer. Then the menu displays several options. You can search the database in several different ways: by magazine title, by author, by subject, by publication date, or by two categories at once.

For example, say you want to list all articles from COMPUTE!. Simply choose option 1 and enter COMPUTE! when prompted for the magazine name. To list all articles by Charles Brannon, choose option 2 and enter BRANNON in response to the author prompt. Once the listing begins, you can pause it by pressing any key, and resume by pressing P.

Fast Filer accepts abbreviations, so it's usually not necessary to type in the entire name. You can abbreviate COMPUTE! as COMPU, for example. However, you must give Fast Filer enough information to distinguish similar names. If the database contains articles by Butterfield and Buncombe, entering BU for the author lists all articles by *both* authors, since both names share those two characters. Entering BUT would distinguish the two names and list all Butterfield articles.

For added flexibility, options 5 and 6 let you search by more than one category at a time. Option 5 provides an *AND* function to find articles that *share* two categories: For instance, to find all COMPUTE! articles written by Charles Brannon, select option 5 and enter 1,2 (be sure to separate the numbers with a comma). Then enter the magazine and author names as prompted.

Option 6 provides an *OR* function to find articles in *either* of two categories. For instance, perhaps you're interested in machine language. With option 6 you could find every article that was categorized under the subject MACHINE LANGUAGE, *or* that was written by Jim Butterfield (who often writes on that subject). The ability to search two categories simultaneously is very powerful.

Easy Data Entry

Of course, no database is useful until it contains some data. Line 1999 of Fast Filer is a template that shows the format for entering data. To enter new data, simply add new lines to Fast Filer, using line numbers higher than 1999. (Lines 2000–2040 are examples which you can modify or delete.)

Every new entry must be in the form of a BASIC line consisting of a line number and a DATA statement, followed by six data items separated by commas. Here is the format:

```
MAGAZINE TITLE,AUTHOR,SUBJECT,
DATE,PAGE NUMBER,COMMENTS
```

Because Fast Filer separates data items with commas, you must not put commas within the data itself. For instance, enter BRANNON C for an author's name, not BRANNON, C.

You cannot omit any of the data items for a particular entry; if you do, the entire list of data is thrown out of sequence. Instead of leaving a particular item blank, substitute something like N/A (for not applicable). For example, you might have an entry for which you don't feel the need to add a comment, like:

```
2000 COMPUTE!,READERS FEED-
BACK,LOWERCASE FOR
TI,4/85,22,N/A
```

Pay particular attention to line 2050, which tells Fast Filer it has reached the end of the data. This must *always* be the last DATA line in the program. When adding new data, renumber this line accordingly. When you're done adding data, resave Fast Filer on disk or tape. The next time you run it, all the new data is available. Since the data is appended to the program itself, the size of the database is limited only by your computer's memory.

Designing A Database

Fast Filer provides the basic framework for a database, but for maximum flexibility, it leaves the most important design choices up to you. You are free to choose whatever

subject categories you like, making them as general or as specific as your needs require.

Creating categories deserves some careful forethought. Clearly, a subject category like COMPUTERS is too broad to be useful. On the other hand, the subject must have enough breadth to encompass more than one article. Consistency is essential, too. If you pick MACHINE LANGUAGE as a subject, then stick with that subject name; categorizing other machine language articles under subject names like ML or MACH LANG will result in incomplete searches.

Before adding your first entry, you may want to decide on standard names for your major categories. These could be saved for future reference in a written list or added to Fast Filer as REM statements.

Use consistent names for magazine titles and authors as well. If you enter a magazine title as COMPUTE (without the exclamation point), it won't be found when you search for articles under the key word COMPUTE! (although the reverse would work). Likewise, GAZETTE is a more convenient title than COMPUTE!'S GAZETTE.

Fast Filer's ability to abbreviate can work to your advantage. For instance, say that you pick GRAPHICS as a major category. If you enter graphics articles under subject names like GRAPHICS VIC, GRAPHICS C64, GRAPHICS PET, and so on, then Fast Filer can find *all* graphics articles (under the subject GRAPHICS) as well as graphics articles for a particular computer.

There are limits to what Fast Filer can do, of course, as there are with any BASIC program this brief. But its simplicity makes the program easier to customize. One of the best ways to learn programming is to begin with an existing program and alter it to fit your own needs. Such changes can range from the purely cosmetic (changing screen or character colors) to more significant improvements (formatting printer output, adding extra categories, etc.). In fact, with only a few modifications, Fast Filer can be used to index practically anything, from books or record albums to investments, rare coins, or stamps.©

Solarpix

Simon Edgeworth

Version by Tim Victor, Editorial Programmer

Planets move in graceful orbits, comets hurtle toward the sun, and the moon turns in its eons-old patterns. This impressive series of astronomical simulations for the Apple II+, IIe, and IIc shows all this, and more. Works in DOS 3.3 or ProDOS. Disk Files: SOLARPIX, SOLARPIX.BIN.

Whether you're new to stargazing or a veteran astronomer, this easy-to-use demonstration of the planets and comets is fun to watch. "Solarpix" is a four-module simulation which includes:

- **Sun and Planets**, simulating the movement of the planets around the sun
- **Sun, Earth, and Moon**, showing how the phases of the moon are created by its orbit
- **Comets**, illustrating the path of Halley's and other comets
- **Facts**, listing the key properties of each planet, the sun, and the moon.

Solarpix consists of two files—a main program written in BASIC, and a binary file containing shape data, character data, a character generator, and a sprite emulator. Press down the Caps Lock key before running Solarpix.

Copernican Motions

In the first module, Sun and Planets, you can watch the orbits of up to four planets at once. The sun is stationary at the center of the screen. Select the innermost and outermost planets you'd like to see, and a year from 1 to 2000. Once the screen is set up, press G to put the planets in motion, the plus key (+) to increase the speed, the minus key (−) to reduce the speed, and S to stop. Elapsed time is displayed in years on the right side of the screen. When you're through, exit the module (and all others) by pressing the Z key.

Lunar Lore

The second module, Sun, Earth, and Moon, shows how the moon's orbit around the earth causes its different phases. The earth is shown orbiting the sun, while the moon orbits the earth. The distance between the earth and the moon has been magnified by 30 times to make it visible. You'll see the current appearance of the moon in the top right corner of the screen. Elapsed time in days is displayed to the right. You can control the motion by using the G, S, +, and − keys as you did in the first module.

The moon takes 27.3 days to orbit the earth. However, it's 29.6 days from one full moon to the next. If you watch the display carefully, you can see why this happens—the moon actually has to complete slightly more than one orbit around the earth to reach the same position relative to the sun.

Because the moon's orbit is slightly tilted, the moon, earth, and sun are normally never exactly aligned. On the rare occasions when they are, an eclipse occurs. If this happens at full moon, we see a

lunar eclipse. If it happens at new moon, we see a solar eclipse.

Dirty Ice

The third module, Comets, shows the orbits of these dirty snowballs. Planets and comets move in elliptical orbits, which look like flattened circles. The extent to which the circle is flattened is called its eccentricity, and ranges from zero (circular) up to almost one. The eccentricities of the major planets vary widely—Venus's is almost zero, for instance, while Pluto's is approximately 0.25. A moving body—say, a comet—with an eccentricity of one or more would leave the solar system.

Enter a value between zero and .96 and press Return to set a comet (real or imaginary) into motion. Notice that the speed of a comet varies as it moves around its orbit. It moves slowest when it's farthest from the sun, and fastest when it's closest to the sun. This is true of planets too, but it's far easier to see in comets' orbits.

Try entering these eccentricities—they represent actual comets.

Faye	.56
Finlay	.70
Enke	.85.
Halley's	.96R

(The R in Halley's stands for reverse, because the comet orbits in the opposite direction of the planets.)

To exit this module, you have to hit Return after you've pressed the Z key.

Facts And Figures

The last module in Solarpix provides some interesting facts about the solar system. From the menu you can select a planet, the sun, or the moon, then view all the facts about it; or you can select a characteristic, such as mass or temperature, and compare the values for the nine planets plus the sun and the moon. Because the distances from the planets to the sun are so hard to visualize, another item, "Drive Time," has been added. It shows the amount of time it would take to drive from a given planet to the sun while traveling at 55 miles per hour. For the moon, the drive time is given for travel from the moon to the earth.

Masses are given in units equivalent to the weight of the earth because the values in tons are too large to comprehend (the earth weighs about 6,588,000,000,000,000,000,000 tons). Densities are compared with water (notice that Saturn is *less* dense than water). Temperatures are average surface temperatures. Moons are listed in order of distance from their planet, starting with the closest. Moons listed as X have not yet been named.

For the amateur astronomer, here are a few technical details. In Solarpix, all orbits are assumed to be on the same plane. For Pluto, however, true elliptical motion is simulated. The planet screen is oriented with Pluto's aphelion on the right, and the positions of the planets are calculated from their actual positions on October 27, 1984. All the graphics screens show views from ecliptic north. ©

Your Personal Ledger

Alan H. Stein

This powerful, but simple-to-use financial application helps you track your expenses, income, assets, liabilities, even your taxable expenditures. You can quickly enter transactions, sort them, and print comprehensive reports. For all Apple II-series computers using either DOS 3.3 or Pro DOS. Disk File: LEDGER.

"Your Personal Ledger" is exceptionally easy to use, yet surprisingly powerful. Each financial transaction can be coded in two different ways, each with a choice of 26 different user-generated codes. Naturally, if a mistake is made, you can edit or even delete transactions. Reports can be printed with records chosen by date and/or code. It's a simple matter to get a list of all expenses for a given period of time, along with a total for those expenses.

Your Personal Ledger is completely written in Applesoft BASIC and is compatible with both ProDOS and DOS 3.3.

Starting Your Personal Ledger

The easiest way to see how to use Your Personal Ledger is to simply go through a sample tutorial session. Let's begin.

Place the disk containing Your Personal Ledger in the disk drive (drive 1 if you have a two-drive system), turn the computer on, and run the program by typing *RUN LEDGER*. The first screen you'll see shows the following display:

```
PRESS    TO
-----    --
1        USE EXISTING LEDGER
2        SET UP NEW LEDGER
```

Since you've never used Your Personal Ledger before, press *2*. You'll be instructed to put a formatted disk in drive 1 and press Return. (If you'd previously used Your Personal Ledger, you would have pressed *1*).

The next thing you'll see is the main menu:

Since you've not used Your Personal Ledger before, it's best to first set up some transaction codes. Press *1*.

Because there are two different sets of transaction codes available, you're asked which you want to edit. Start with code 1, and you'll see the following display:

```
EDIT TYPE 1 TRANSACTION CODE

CODE DESCRIPTION

A

PRESS     <--OR-->     FOR OTHER
                       CODES
          <C>          TO CHANGE
          <ESC>        FOR MENU
```

Each transaction code type has 26 codes, A–Z, each of which has a description. Since you've not assigned any descriptions yet, the DESCRIPTION column is blank. Let's assign some type 1 codes. Almost everyone will find categories for *income, expenses, assets, liabilities,* and *other.* These descriptions can be changed at anytime, but it's a good idea to do some advance planning.

To assign the description INCOME to code A, press C (for change) and then type *INCOME*. Of course, press Return after entering *INCOME* to let your Apple know when you're done. Your Personal Ledger generally doesn't require a press of the Return key for single key responses, but it does need it when the response can be of an unknown length.

After typing in the description, you'll see it displayed where only a blank space appeared before. Now, assign the description EXPENSE to code B. That's almost as simple—just press the right arrow key and *B* appears beneath under CODE. You can then type in the description *EXPENSE*. In a similar way, assign ASSET to code C and LIABILITY to code D. But what if you want to assign OTHER to code Z? You can display code Z either by pressing the right arrow key several times, thus going through the alphabet, or by pressing the left arrow key a few times and backing up (using the left arrow key shows Z immediately after *A*). When you're finished, press Esc to return to the main menu.

So far, you've just set up a few

type 1 codes. Now for type 2 codes (you can think of type 2 codes as subcodes of a sort). Again from the main menu, choose EDIT TRANSACTION CODES, this time picking type 2 codes. They're entered the same way as type 1 codes. To start with, enter descriptions of *salary, interest income, rent, food, utilities, entertainment* and, naturally, *other.*

Transactions

You're now ready to enter some transactions—press 2 from the main menu. You'll see prompts on the screen for a date, check number, codes, source/payee, description, and amount.

The date must be entered in the form *mm/dd/yy.* The slashes are supplied automatically and the Return key does *not* have to be pressed. However, all single-digit numbers must be entered with leading zeros. For instance, June 5, 1986 would be entered by typing *060586,* and displayed as *06/05/86.*

The check number can either be entered or omitted by pressing Return. You can also enter non-numerical information here—for example, a MasterCard purchase might be entered with *MC* in the check number column. Numbers longer than five digits are truncated, leaving only the *last* five digits. If you're entering text here, the same restriction applies.

The codes are entered in a manner similar to the way they were edited—press the arrow keys until the desired code comes into view, then press Return.

Source/Payee represents either the source of income or the person or company a payment is made to. For example, if you received a check from *COMPUTE!'s Apple Application,* you could type *COMPUTE,* while if you paid for your groceries by check, you'd enter the name of the market. Your Personal Ledger allows 20 characters for this entry.

Another 20 characters is allowed for the description, which might be *March groceries* or *sub to COMPUTE!.*

Finally, the amount would be entered, using only numbers and a decimal point, not the dollar sign ($).

After entering each transac-

How Your Personal Ledger Works

Your Personal Ledger stores all transactions in an array called RD$. Each element in the array corresponds to a transaction. The first six characters represent the date, stored in the form *yymmdd.* The next five characters contain the check number. The following two characters contain the type 1 and type 2 codes (a memory saving technique). Twenty characters each are set aside for the source/payee and description, with the last nine characters representing the amount of the transaction.

The actual descriptions of the codes are stored in an array CODE$.

When the program executes, these code descriptions along with all the transaction date are read from the sequential disk file LEDGER .DATA. The actual number of transactions is also stored in that file and is assigned to the variable NR. All this is performed in lines 380–510. (When setting up a new file, this step is skipped. All elements of CODE$ and RD$ are automatically initialized to null strings and NT is initialized to 0.) A pointer to the highest element of the RD$ array, TR, is initialized to the same value as NR.

Lines 750–1060 perform the assignment of descriptions to the transaction codes by simply setting elements of the array CODE$ to the description entered.

Lines 1070–1240 control the entry of new transactions. Separate subroutines are used for each field. When the entry is complete, the number of records is incremented as well as the pointer to the top record in the array RD$. Lines 4150–4180 combine the different fields into a single character string RD$, which is then stored in the array RD$.

A variable RN is used to point to the last transaction accessed. When the option FIND/CHANGE TRANSACTION is chosen, the transaction pointed to by RN is retrieved and separated into individual fields by the subroutine on lines 4190–4270. If another transaction is desired, the value of RN is increased or decreased and the process is repeated. If the transaction is to be changed, the same subroutines used for entry are used to change the fields.

When a transaction is deleted, the corresponding entry in the array RD$ is set to the null character string and NR is decremented. This is performed by lines 1810–1920.

Sorting of transactions is performed in lines 2030–2210. A fairly straightforward procedure similar to a bubble sort is used. Each element of the array RD$ is processed in turn. The element being processed is simply compared to each of those already processed and swapped until it has reached its proper place relative to the elements already processed. Deleted entries are easily recognized since they are stored as null strings and are skipped over. When the process is complete, the pointer TR is reset to the same value as NR.

Although this sort process is relatively primitive, it's fairly effective in this situation since most of the array will already be in order.

The printing of a report (lines 2220–3190) is straightforward. Each transaction is printed, with the length of each field determined by the value of a variable set either when the program started or when the user decided to determine a custom format. If minimum or maximum dates were used, any record not falling within that time span is ignored. Similarly, if either code type was specified, any record with a different code is ignored. The total amounts of all the transactions printed are stored in a variable A. When all the transactions have been printed, the value of A is then printed as well after some formatting.

Lines 3200–3410 control the saving of the transactions to disk and exiting the program. All the elements of the array CODE$ are written to disk, followed by the value of NR (the number of records) and then the non-blank elements of the array RD$.

Variable	Function
A1	Temporary use
A$	Input and temporary storage
B0$	Null string
B1$	Blank string
B20$	String of 20 spaces
BS$	Backspace character
CLRLN	Location of routine to clear line
CODE$	Array storing code descriptions
CS	Location of routine to clear screen from cursor
D$	CTRL-D
ESC$	Escape key
FS$	Forward space
I,J	Counters, indices
LD,LC,L1,L2,LP,LS,LM	Field lengths used for printing reports
MR	Maximum number of transactions
NC	Number of codes −1
NR	Number of transactions
QU$	Quote
R$	Return key
RD$	Current transaction
RD$(*)	Array storing transactions
RN	Pointer to transaction
TR	Pointer to top transaction
T$,Z$	Temporary storage
DT$,CK$,CD(*),PY$,DC$,AMT$	Fields of current record

tion, you're given the choice of pressing the Esc key to abort the transaction (and thus return to the menu) or pressing Return to accept the transaction. If you accept the transaction, you're taken automatically to the FIND/CHANGE menu, which can also be reached directly from the main menu (option 3).

You can easily view other transactions (assuming you've entered more than just one) by pressing either arrow key or by pressing the *J* key. Pressing an arrow key moves forward or backward by one transaction, while pressing *J* gives you the opportunity to move quickly through the file by specifying a number of transactions to move. Entering a positive number moves forward, while entering a negative number moves backward in the file. (If you want to move forward, you *don't* need to put a plus symbol (+) before the number of transactions to jump.)

Pressing *P* prints the transaction information, if you have a printer connected properly. Pressing *D* lets you delete the transaction. By this time, you probably haven't entered any transactions you want to delete, but keep this option in mind for later—you'll certainly find a use for it.

Pressing *C* gives you a chance to change any of the information in a transaction. You're led through each transaction and given the opportunity to either re-enter the information—in the same manner it was originally entered— or leaving it as is.

Hitting *E* lets you enter another transaction, without returning to the main menu. Press the Escape key to get back to the main menu.

Now Sorting, Now Reporting

The fourth choice from the main menu, SORT TRANSACTIONS, rearranges the transactions in chronological order. It also compresses the information in memory. Your Personal Ledger is set up for a maximum of 400 transactions. However, transactions deleted in a session count towards that maximum until the next session, *unless* the transactions are sorted. Thus, if you're unable to enter more transactions, you may be able to make room simply by sorting.

Option 5 is PRINT REPORT, the facility which enables you to get a hard copy of the data you've put in your file.

When you print a report, the first thing you're asked is whether you want to use the default format. This format prints all fields at a predetermined width. If you decide to devise your own format, you're shown the default field widths for each field. You can accept each default by simply pressing Return, or change it by pressing *C* and entering the new width. If you want to omit a field from the report, define its width as zero. (Any custom format becomes the default format for the rest of the session.)

Let's say that you want to print a listing of all your utility bills—you won't need the second code type printed on this report. Define your own format by accepting all defaults except for the type 2 code, which you should change to a width of zero.

After determining the format, you must determine which records are to be included. The first display you'll see looks like:

```
SELECTION
PRESS <RETURN> TO ACCEPT ALL
  DATES
OR
PRESS <C> TO CHOOSE PARTICU-
  LAR DATES
```

If you press Return, the date of each transaction will not be a factor in determining whether to include it. If you press *C*, then you'll be asked for a time span—that is, an earliest date and a latest date. Provide the month and year of these dates in the form of two digits for the month and two digits for the year (*0586* could be an example).

Next, you'll need to specify which codes to accept for each type (1 and 2) code. In each case, you can choose to include records with any code by simply pressing Return, or choose records which have a particular code by specifying it.

Since you want a listing of utility bills, you'll press Return for each selection criterion, except for the type 2 code, which you should specify as UTILITY.

Once the selection process is complete, Your Personal Ledger prints the report, complete with a total at the bottom, and goes back to the main menu.

Saving Grace

When you're finished, press the Esc key when the main menu is displayed. This selects the SAVE CHANGES/EXIT option. This is how you exit Your Personal Ledger, or how you periodically back up your data file during a long session. (You have one last chance to go back to the menu without saving

the changes by pressing Esc again.) Pressing Return saves the changes to disk. At this point, you can re-enter Your Personal Ledger by hitting the Esc key, or end the session entirely by again pressing the Return key.

More Personal

Your Personal Ledger is so simple, yet so powerful, that there's probably little you might want to change. It's set up for a maximum of 400 transactions in order to avoid running out of memory. This is a conservative figure chosen to avoid having to either carefully calculate how much RAM was available for DATA, or to put in sophisticated error detection code to avoid termination of the program if it ran out of memory. That maximum can easily be changed (change the value assigned to MR in line 190) and is an option if your Apple has additional memory and an enhanced version of Applesoft BASIC able to handle that memory.

If your Apple II computer doesn't have this capability, you can still record more than 400 transactions, though this requires additional disks. For instance, you may have to keep two (or more) data disks available. Simply label each disk with the appropriate time frame, say *January–June 1986.* If you have a multitude of transactions to record, you may even have to go to a one-disk-per-month system. (You have to use additional disks since Your Personal Ledger always saves its data to a file called LEDGER .DATA.)

In addition, you may be interested in a graphical depiction of your financial status. The system used to store the data on disk is simple enough so that it would not be difficult to extract the data in a form usable by a standalone graphics program (such as *Visiplot/Visitrend*) or a simple program you might write.

Your Personal Ledger is an easy to use, yet flexible system for keeping track of your financial situation. Indeed, if you'd spent the time it took to read this article using Your Personal Ledger, you'd already have figured out how much you went over budget last month! ©

Apple SuperFont Custom Character Set Graphics

Tim Victor, Editorial Programmer

Here's a significant enhancement for graphics on Apple II-family computers. With "Apple SuperFont," you can now place upper- and lowercase text anywhere on the high-resolution screen. In addition, you're not limited to the built-in character set, either—you can easily define foreign character sets, italics, boldface, and underline fonts, as well as shapes for high-speed animated games in BASIC. The program requires a 48K or 64K Apple II+, Apple IIe, or Apple IIc, with either DOS 3.3 or ProDOS. Disk Files: APPLEFONT, APPLEFONT3.3, APPLEFONT2, HROUT, NORMAL .SET.

Without resorting to machine language, programming high-speed graphics is difficult on the Apple. High-resolution graphics look nice, but shape tables are too slow for most animation purposes. One alternative is to use character graphics for animation. Characters can move a whole block (character position) at a time, and can be placed on the screen with a simple PRINT statement. Unfortunately, ordinary Apple characters aren't very suitable for games or even business charts.

But now there's a way around these problems. With "Apple SuperFont" and its accompanying utility programs, you can easily redefine a character into practically any shape you want and print it directly on the hi-res graphics screen. Custom character sets are a snap to design, and fast animation is as simple as printing a character, erasing it, and printing it again in a new location.

Several programs already exist for printing characters on the hi-res screen, including HRCG (High Resolution Character Generator), which is part of the Apple DOS Toolkit. The Apple SuperFont HROUT program works much like HRCG, putting characters on the high-resolution screen from a table of character images, but the Apple SuperFont system is much more versatile.

The Apple SuperFont Editor makes it easy for you to create character sets (fonts) for use with HRCG or HROUT. Special features help you design multicharacter shapes and allow you to see the effects of the Apple's unusual use of color in hi-res graphics. Once you've creat-

ed or customized a character set, you can easily use these fonts in your own programs.

Typing Apple SuperFont

To run SuperFont, you need to have four files on the same disk: APPLEFONT, APPLEFONT2, HROUT, and NORMAL.SET. There are two different versions of APPLEFONT. Use APPLEFONT itself for ProDOS, and APPLEFONT3.3 with DOS 3.3. The other three files need no changes to be used with either disk operating system.

When you're ready to animate, type RUN APPLEFONT (or APPLEFONT3.3). APPLEFONT first checks to see which operating system is in your Apple. If the correct operating system for this version of APPLEFONT is present, it will BLOAD the other three files, and connect HROUT to the standard character output routine. APPLEFONT2, the SuperFont Editor, is started with a CALL to 4096. From then on, the SuperFont Editor is in complete control except when it needs to access the disk drive. If you ask to load or save a character set, control returns to the BASIC program, the file is transferred using BASIC's disk access commands, and the SuperFont Editor program is CALLed again.

Using The SuperFont Editor

Characters are designed and edited on a grid that represents 32 (vertical) × 55 (horizontal) pixels. Each cell in the grid is a fourfold enlargement of actual size. Individual cells can be turned on (white) or off (black) with the bit-editing functions, and blocks of cells can be copied from one place to another on the screen. Patterns of 7 × 8 cells can be saved from the screen to the character set being edited with the Put command. The Get command does just the reverse, pulling a character from the character set onto the editing screen.

All of the features of the Editor are controlled with a series of four menus, entitled Bit Edit, Charsets, Utility, and Display. Each of these menus contains three to six selections. Only one menu is displayed on the screen at a time.

To change menus, press the space bar. The next menu title will be printed on the screen, along with its menu selections. The top selection will be printed in inverse characters to indicate that it has been chosen. To select a different menu item, use the left- and right-arrow keys. The large cursor bar moves up or down the menu to show you which selection is active.

Some menu items, like Clear Screen or Save Set, wait for you to press the RETURN key before performing their functions.

Three Cursors

You will be using three visually distinctive cursors in the SuperFont Editor: the bit cursor, the box cursor, and the character cursor. When a menu item is selected, one of the cursors may begin to flash, indicating that it can be moved. The cursors are controlled by a keypad centered on the D key:

The bit cursor is a 1 × 1 cell box displayed on the editing screen. It flashes whenever the Bit Edit menu is displayed. Moving the bit cursor around on the editing screen sets (white) or clears (black) the cells that the cursor passes over. In other words, the bit cursor leaves a trail of black or white behind it. Selecting Black or White changes the color drawn when the bit cursor is moved. If you want to move the bit cursor without drawing on the screen, select the Move option.

The box cursor is a box displayed on the editing screen, but its size can be changed. It can be as small as a 1 × 1 cell, or as large as the entire editing screen. When you're using a utility such as Copy or Flip, the box cursor outlines the area on which the utility will operate. These utilities can be used on a character, part of a character, on shapes made up of several characters, or on a portion of a character, simply by changing the size of the box. Pressing the RETURN key when Flip is selected turns the contents of the box cursor upside down, and the Mirror function reverses left and right sides of the box. The Invert function changes all of the white cells inside the box to black cells, and all black cells to white. When Copy is selected, the cursor pad controls a second box cursor, which initially appears on top of the original box. Pressing the RETURN key copies the contents of the original box to the second box.

You can also use the box cursor to select the 7 × 8 cell character pattern for the Put and Get functions. The character cursor, located in the character set displayed at the bottom of the screen, flashes when the Get or Put function is selected. Use it to select the character that is the source of the Get or the destination of the Put.

The contents of the box cursor are displayed at actual size (one cell = one pixel) in the upper-right corner of the screen. Two parameters, HB and PX, affect how colors are presented. Pressing the RETURN key when the High Bit menu entry is selected changes the setting of HB. In Apple hi-res graphics, the status of seven one-bit pixels is stored in the lower seven bits of a byte in memory. The eighth bit, the most significant bit, controls the colors in which these bits will be drawn. When drawing on the high-resolution screen in BASIC, the high bit is clear when HCOLOR is between zero and three, and is set when HCOLOR is between four and seven. The display is in blue and orange when the high bit is set, or green and violet when the high bit is clear.

The Even/Odd menu entry controls whether this display starts on an even or an odd pixel (PX). When a shape is shifted by one bit, the colors in the display are reversed (blue for orange or green for violet). The alignment of the shape is changed by pressing RETURN when Even/Odd is selected.

At the bottom of the screen, all of the characters in a 96-character set are shown. With the RAM/ROM function in the Display menu, the character set displayed can be either the set you are currently editing or the hardware character set in your Apple. Get and Put operate only on the RAM character set no matter which set is being displayed.

HROUT, The Character Generator

Apple SuperFont uses a machine language graphics utility called HROUT, for high-resolution output. HROUT links into the standard character output vector and permits

text to be displayed on either hi-res screen. Because the standard text output routine also remains active, the PRINT command, and any other text commands, can be used to create hi-res text. HROUT's only limitation is that it cannot perform screen scrolls at the bottom of the screen.

To use HROUT in your own programs, BLOAD it into memory. It can be loaded anywhere in memory, but to make things simpler, we'll use location $300. First, let HROUT know which character set to use by POKEing the address of the character set into locations 6 and 7, low byte first. If you put your character set at $8D00, the POKEs are:

```
POKE 6,0 : POKE 7,141
```

If you are using DOS 3.3, you can activate HROUT by entering:

```
POKE 54,0 : POKE 55,3 : CALL 1002
```

When in immediate mode, these commands have to be entered together on a multistatement line (separated by colons). They can be on separate lines in a BASIC program, but the three commands should be executed one after another. Since locations 54 and 55 are being POKEd with the low and high bytes of the address of HROUT, these POKEs will be different if you put HROUT somewhere other than $300.

From ProDOS, it's easier to turn on HROUT. Just type

```
PR# A$300
```

Avoiding Screen Scrolls

Since HROUT concludes by calling the standard ROM routine for displaying a character on the text screen, all cursor control remains the same. You can move to any location on the screen by using the HTAB and VTAB commands. HOME still moves the cursor to the upper left of the screen, but will not clear the hi-res screen. To get the equivalent of a text HOME, use HOME : CALL −3092. The routine at −3092 clears the current hi-res screen and turns on hi-res graphics.

If you need to know what's where on the screen, you can PEEK to the text screen. By taking a couple of precautions, both text and hi-res screens should be the same. First of all, make sure that you clear both screens at the same time, as mentioned above. Second, don't let the text screen scroll. In order to make HROUT as small (88 bytes) and fast as possible, no provision was made for scrolling the screen. This could even be to your advantage for many applications, but you have to be careful if you want the text and graphics screens to agree.

The biggest problem arises when you print to the last character on the twenty-fourth line. Even if you follow the PRINT statement with a semicolon, the cursor will wrap onto the twenty-fifth line and the screen will scroll. There is a solution: Fool the computer into thinking the screen has 25 lines by using POKE 35,25. The output routine will then have no qualms at all about advancing the cursor to the twenty-fifth line, leaving it there, and even printing there. A lot of responsibility now rests on your shoulders, because the twenty-fifth line doesn't really exist. Printing something there is the same thing as POKEing out of the range of the text screen. That could cause significant problems.

If you change the text attribute with the INVERSE or FLASH commands, the bit patterns will be reversed before they are plotted on the screen, inverting the character. The NORMAL command also works, canceling inverted printing.

Now you can label high-resolution charts and graphs with a choice of any font, and you can design these fonts yourself with the Apple SuperFont Editor. Also, letters of the alphabet can become detailed shapes, permitting fast high-resolution game graphics in BASIC. In fact, we've started using this technique ourselves for some of the Apple games published in COMPUTE!. ©

Apple Animator

Steve Johnson
Version By Tim Victor, Editorial Programmer

This feature-packed utility makes it a breeze to create your own short cartoons or animation sequences on any Apple II-series computer screen, using either DOS 3.3 or ProDOS. Disk Files: ANIMATOR, ANIMATOR2.

Computer animation can be marvelous to behold, but a drudge to produce. Whether you're working in BASIC or machine language, creating objects and manipulating them on the screen can mean fumbling for hours with PEEKs, POKEs, bits, bytes, and other tedious details.

"Apple Animator" goes a long way toward automating this process. It works much like a cartoonist's sketch pad, letting you draw a series of similar images which are then displayed in rapid sequence to create the illusion of movement. Your finished cartoons can even be saved on disk and reloaded for viewing later.

Apple Animator requires two files on disk: the main BASIC program (ANIMATOR) and a binary file (ANIMATOR2) that contains graphics data.

Drawing An Image

When you run Apple Animator, it displays an editing screen with 20 numbered frames. You can draw as many as 20 pictures, one in each frame, then flip rapidly through the frames to create animation. The frame number displayed at the upper left of the screen shows which frame you're currently working on. Normally, Apple Animator begins the animation with frame 1 and ends with frame 20. But you can start and end the animation wherever you like. For example, a short sequence might start with frame 1 and end with frame 3. To view only part of a long sequence, you might start at frame 12 and end at frame 18. The frame number is selected by pressing the right- and left-arrow keys.

The frame number also determines which frame you'll be working on when you go to the editing screen. Let's start with a simple example. Make sure the frame number is set to 1; then press the 0 key to select the editing function and press Return at the next prompt. After a brief pause, Apple Animator displays a drawing grid with a blinking cursor. Edit mode has three main functions and several secondary features. Each one is chosen by pressing a different key.

Key	Function
1	Quit
2	Draw
3	Move
4	Erase
5	Clear
6	Save
7	Invert
8	Update
9	Revert

The functions you'll use most often are Draw (2), Move (3), and Erase (4). Notice that when you first enter the editing screen Move is already selected, as indicated by the bullet symbol (•) beside it.

For now, concentrate on just these three features. *Draw* draws wherever the cursor moves, *Erase* erases wherever the cursor happens to be, and *Move* lets you move the cursor without disturbing anything on the screen.

When the editing screen appears, move the blinking cursor (which first show in the upper-left corner of the grid) left, right, up, or down by pressing the J, L, I, and K keys, respectively.

Draw a simple shape on the grid to become familiar with these basic functions. Though Apple Animator does not display the shape in

actual size while you draw, you can see it displayed by pressing the 8 (Update) key. The shape appears on the right. An invert function (7) lets you reverse everything on the grid—every dot becomes a blank, and vice versa. Be patient—it takes Apple Animator awhile to complete this process.

Once the picture is finished, you can press 6 to save it and return to the main screen. *Note that you must save a picture by pressing the 6 (Save) key to put it in the frame.* If you exit the edit mode by pressing 1 (Quit), the new picture is lost and Apple Animator uses whatever that frame previously contained. Try drawing a simple shape and saving it. Since this is just for practice, any scribble will do. When you return to the main screen, Apple Animator displays the picture in frame 1.

Frame By Frame

Now you're ready to draw the next frame in the sequence. In most cases you'll want to make only slight changes from one frame to the next, to simulate smooth motion. To save time, Apple Animator lets you copy a picture from one frame to another. Let's demonstrate this by copying the picture from frame 1 to frame 2. Set the picture number to 1 with the arrow keys, then press 0 to edit. Apple Animator displays a prompt, inviting you to enter a frame number. To edit the current picture number, you would just press Return. However, by entering a *different* number you can copy the current picture into a different frame, then change that picture to make the next frame in your cartoon.

When you enter 2 (for frame 2) at the prompt, Animator copies the picture from frame 1 into the drawing grid. When the drawing grid appears, make some change in the picture to distinguish it from frame 1. Now press 6 to save the picture in frame 2 and return to the main screen. Apple Animator displays both pictures in their respective frames.

After drawing a few frames, you're ready to bring them to life. The first step is to specify the starting and ending frame numbers. The starting number determines which frame begins the animation, and the ending number tells Apple Animator where the series ends.

Set the starting number first. Use the arrow keys to set the frame number to 1, then press the 3 key. Now use the arrow keys to make the frame number match the *last* frame that contains a picture—press the 4 key. This sets the ending number. You must always set the starting and ending numbers before selecting animation (if you don't, Apple Animator flips through all 20 frames whether they contain pictures or not). Once these numbers are set, press the 5 key to view the animation sequence. Hit the space bar to pause and Return to stop.

By selecting different *speed* and *pause* values, you can move the animated figure across the screen. The speed value can range from −15 to 15. When it's 0, the figure is animated in place. Positive values move the figure from left to right; negative values move it from right to left. The greater the value, the faster the figure moves. Press the 6 key to increase the animation speed, and 7 to decrease it.

The pause value controls the time delay between each frame of the animation. A small pause value makes the pictures change very quickly, while larger values slow down the process. Pressing the 8 key selects a smaller pause; the 9 key selects a a larger value.

Macro Editing Features

Apple Animator provides a few macro (large-scale) editing features to help you work with longer cartoons. All these features are accessible only from the main screen. The insert function lets you insert a blank frame anywhere in the series. To use it, set the frame number to the number of the frame where you want to insert a blank, then press the 1 key. The designated picture and all those following it are bumped forward one frame. When you insert, the picture in frame 20 is always lost.

The delete function lets you delete any frame in the series. Change the picture number to the frame you want to eliminate, then press 2. All the higher numbered pictures move down one frame, deleting the picture in the designated frame. Frame 20 is always blank after a deletion.

The invert function (press D) works just like invert in editing mode, but inverts all 20 frames at once.

To clear all 20 frames, press E to quit or C to clear. Since these last two functions can have drastic results, Animator lets you abort either one without causing harm.

When you finish a sequence, press B to save it on disk. The screen clears and displays two options: You can press Esc to cancel the save, or Return to list the picture files on that disk. If you simply want to save the animation sequence, type in a picture filename. Since Apple Animator doesn't add a filename suffix for you, it's a good idea to append .ANI to the end of filenames yourself. This makes it much easier to tell which files on a disk are picture files.

Other Editing Features

Besides the primary editing functions (Draw, Move, and Erase), Apple Animator provides a number of other useful features. *Clear*, selected by pressing the 5 key, clears the screen, giving you an empty frame. Additional choices, however, are offered in a menu near the bottom of the screen. If you just want the frame to remain black, press 0 and Return. Notice, though, that other colors are available, from Green (1) to White2 (7). Clearing the frame and choosing a color creates the pixel combinations which represent that color. You can draw on it as usual and, after saving the frame, it will appear in the selected color on the main screen. (Of course, you'll be able to see this only if you have a color monitor or television set connected to your Apple.)

As mentioned earlier, choose *Update* by hitting the 8 key. Almost immediately you'll see an actual-scale reproduction of the shape displayed on the right of the screen. You can use this feature to check the shape from time to time as you're drawing or editing.

Revert, selected with the 9 key, returns the shape to what it was the last time it was saved from the editing screen (not from the disk file). This can come in handy when you've been editing a frame, but decide you like it better the way it was before. ©

Softsearcher

Ilan Reuben

Here's a short, handy, and fast programming utility written entirely in machine language. With it, you can instantly locate key statements and phrases in your programs. It works on any Apple with at least 48K RAM and a disk drive. Disk File: SEARCHER, SEARCHER.SETUP.

Many BASIC programs are constructed and debugged by adding new sections and routines to existing sections and routines. As a result, programs can become long and complex. Debugging becomes a real mess when you have to sift through 2000 lines of BASIC to find a certain routine or statement.

"Softsearcher" is a machine language utility which scans any BASIC program for all references to a specified character or phrase, and tells you where each reference is—all in the blink of an eye. The machine language program itself is just over a page (256 bytes) in length, and resides at memory location 36864 ($9000 in hexadecimal). If you know little or nothing about machine language, don't worry; you can use Softsearcher as long as you can type in a BASIC program and follow a few simple directions.

Using Softsearcher

First, let's get Softsearcher up and running. Load the program from disk by typing:

```
BLOAD SEARCHER
```

Once you have it in memory, you must set the & vector to the start of the program. This lets you run Softsearcher every time you type the character &. From BASIC, enter:

```
POKE 1014,0: POKE 1015,144
```

or from the monitor, type:

```
3F6:0 90
```

Softsearcher should now be ready to use. Here's a sample BASIC program to show how it works. Just type this in (it's not necessary that you run it, although that won't affect Softsearcher's functions).

```
10 PRINT "This is a test"
20 FOR A = 1 TO 10
30 PRINT A + 10
40 NEXT A
```

Suppose you want to find all the references to the variable A in the program. You would type:

```
& A
```

and the computer would respond with:

```
FOUND AT LINE 10
FOUND AT LINE 20
FOUND AT LINE 30
FOUND AT LINE 40
```

To find all the lines in which the number 10 appears, enter:

```
& 10
```

Softsearcher hunts through the program and reports:

```
FOUND AT LINE 20
FOUND AT LINE 30
```

Notice that line 10 was not included, even though there's a 10 in its line number. Softsearcher ignores line numbers. Note, too, that if you're using Softsearcher on the Apple IIc, it will find characters in lowercase as well as in uppercase. It doesn't matter whether the search characters or phrase is in uppercase or lowercase.

Selective Searching

To specify a range of lines for Softsearcher to look through, type the # character after the &, along with the starting and ending line numbers and the phrase to search for. It could look like this:

```
& #20,30,PRINT
```

This searches lines 20 through 30 for the word *PRINT*. You'll see the report:

```
FOUND AT LINE 30
```

Softsearcher can be used *only in direct mode*, not in deferred mode (that is, you cannot call it from a BASIC program). If you try, the message ?NOT DEFERRED COMMAND ERROR appears.

There are some things you cannot search for. If you enter:

```
&&
```

nothing will appear, even if the & symbol was used in a PRINT statement. You also cannot search for the # symbol. If you try, you'll get a SYNTAX ERROR message.

If you'd like to have Softsearcher ready to use every time you boot your system, use the BASIC setup routine SEARCHER .SETUP as a hello program when initializing disks. Just make sure that you've got the machine language for Softsearcher saved on that disk. It must have been saved as SEARCHER. If it wasn't, make sure that you change lines 20 and 50 in SEARCHER.SETUP to match the new name you've given it. ©

Chaining Programs In Applesoft BASIC

Richard J. Kaufman

By appending a few lines to Applesoft BASIC programs, you can cause them to call each other and still keep their variables intact. Disk Files: FIRST, SECOND.

When a BASIC program gets so large that it exhausts all available memory, you have very few options. If you try to make the program more memory-efficient, the first step is usually to delete the REMarks; but this can prove a disaster when you come back to the program six months later. Even the best programmers forget the details of old programs, and can benefit from a few reminders. You can also split the program in two and save all the shared variables in a disk file, which the second program can read and use. This is slow at best. The old DOS manual shows a CHAIN command, which is just what you need in these situations. Unfortunately, CHAIN works only with Integer BASIC, which is no longer supported by Apple. Most high-level languages provide some means to call another program that does not remain memory resident, and include a means of passing variables. Applesoft BASIC, alas, does not. It's too basic.

There is a way, however, to chain (transfer control between) Applesoft programs, leaving the first program's variables and arrays intact. It requires an understanding of how Applesoft handles memory, and a little experimentation, but it will greatly increase the amount of BASIC code you can run at one session. Even if memory size is not a problem, you may find several small, modular programs easier to deal with than one huge one.

Table 1: Applesoft Pointer Locations

Pointer Location		Name	Description
Hex	Decimal		
$67 $68	103 104	TXTTAB	Start of program (usually $0801).
$69 $6A	105 106	VARTAB	Start of simple variable space.
$6B $6C	107 108	ARYTAB	Start of array space.
$6D $6E	109 110	STREND	End of variable space.
$6F $70	111 112	FRETOP	Start of string space.

Inside Applesoft BASIC

Table 1 summarizes the pairs of memory locations that tell BASIC where to find the program, its variables, arrays, and strings.

VARTAB, the start of simple variable space, usually begins right after the program text itself. However, it can be set to any desired value using the LOMEM command. This is the key to our chaining technique. Imagine that a program—which we'll call FIRST—has issued a command to run a second program named SECOND:

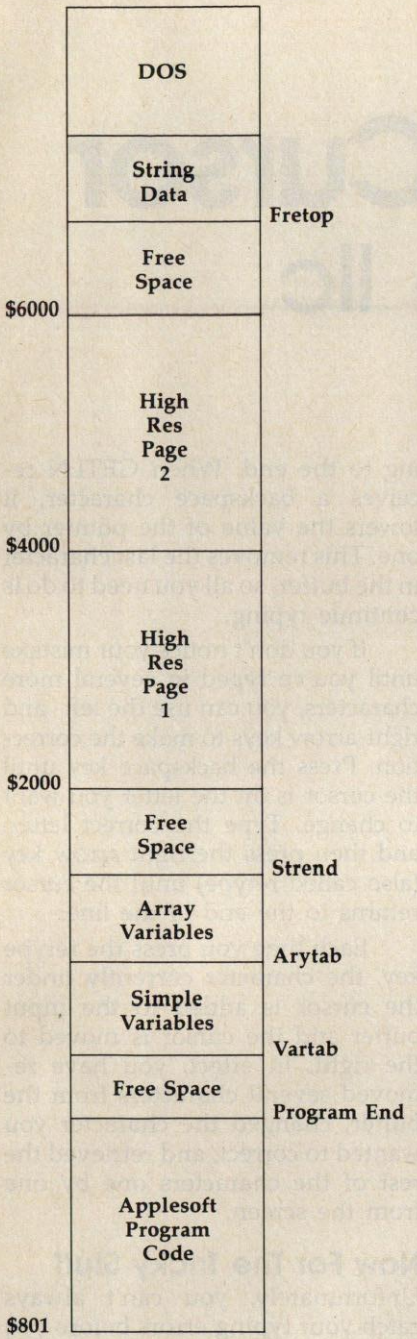
```
PRINT CHR$(4);"RUN SECOND"
```

When SECOND loads at TXTTAB (location $0801), it overwrites the previous contents of BASIC program memory, including at least part of FIRST. Assume that the program text for SECOND is shorter than the text of FIRST. In this case, the variable space is not yet destroyed, and it will not be until SECOND starts to use its variables. The figure helps you to visualize what is happening.

If SECOND sets VARTAB, ARYTAB, STREND, and FRETOP to the values in effect when FIRST was exited, all of FIRST's variables will be available to SECOND. The only exceptions are strings that were assigned values by literal statements in FIRST—for example, A$="THIS IS A STRING"—rather than by an INPUT statement, a disk read, or another string operation. Such strings must be reassigned. This is because of the way Applesoft handles string variables. The entry for a string variable in the variable table doesn't actually contain the text of the string. Instead, it contains a pointer holding the address of the string's text. The text of strings assigned by INPUT or created by some string operation will be stored in the tables above FRETOP. However, for strings with literal assignments, the variable table entry will point to the literal statement in the program text. When another program is chained, the program text containing the literal definition will be overwritten, but the variable table entry will still point into the program text area. Thus, the string assignment is no longer valid.

There is a way to avoid this problem. Literal strings involved in

Applesoft Memory Organization



string operations such as concatenation are given definitions in the string pool above FRETOP, even if the string isn't actually changed. Thus, if you replace the literal string definition A$="THIS IS A STRING" with A$="THIS IS A STRING" + "", the string definition will be preserved after chaining. Adding (concatenating) an empty (null) string to the literal assignment will cause a copy of the string definition to be created in the string pool above FRETOP.

Implementing An Applesoft Chain

The need to chain BASIC programs often is brought about by the use of high-resolution graphics. Program space starts at $0801, and hi-res graphics page 1 starts right in the middle of the program space, at $2000. Translated to decimal, this leaves a mere 6144 bytes for the program. You can use hi-res page 2 for graphics (it starts at $4000), but suppose you need both high-resolution screens for some special effects? The program FIRST shows how to cause an Applesoft program to load at any desired location in memory. (Starting the program above hi-res page 2 will give you more room, unless you use many large arrays or lots of strings.)

Line 5 of FIRST is the key to this technique. This line can be appended at the beginning of any program you wish. Note that the variable LOC is set to the one location beyond the last byte of hi-res screen 2. This is fine if you want the program to load at that location. Otherwise, set LOC to the address of the location where you want your program to start. Of course, if you're not doing graphics, you can leave the program loading address alone.

Line 6 sets LOMEM, the bottom of variable storage, to location 34817. This gives you 10,241 bytes of program text space above hi-res screen 2. The optimum value for LOMEM will be determined by the longest program to be chained.

To calculate the approximate length of a program, multiply the number of sectors shown in the catalog entry for the program by 256 for DOS 3.3 or by 512 for ProDOS. For instance, a program that occupies four sectors when saved under DOS 3.3 is about 1024 (4 * 256) bytes long. A four-sector program saved under ProDOS will be about 2048 (4 * 512) bytes long. A more precise method is to check the difference between the values of VARTAB and TXTTAB after loading the longest program in the chain and use that value. The following line shows how to do this:

```
PRINT (PEEK(105) + 256 * PEEK(106))
  -(PEEK(103) + 256 * PEEK(104))
```

Once you've determined the program length, add that value to the program load address to get the minimum value for LOMEM.

In practice, it's usually easiest to start with 35000 for LOMEM and adjust the value if you find that you have memory problems. If you are not using both graphics screens, you can use a much lower value for LOMEM, but this is not likely to be necessary unless you are using a lot of array space.

Lines 10–30 of FIRST assign a literal value to a string, use that string in a concatenation operation, print it, accept a numeric value for the variable X, and assign the value 9999 to Y.

We now are ready to save the program pointers, a task which is done in lines 900–910. The top of hi-res page 1 is 16383. We work down from there, putting the pointer bytes into the top of the screen. They do not show up on the screen, since they are nondisplayed "slack bytes." If you are not doing graphics, and you need to chain, you may have to find some other safe place to store your pointers. Perhaps you can use space in one of the DOS buffers or else find some unused space in low memory. If waiting for the extra time required doesn't bother you, the pointers can even be saved to a file. Line 920 runs the next program, SECOND.

Lines 5 and 6 of SECOND retrieve the program pointers stashed by FIRST. Lines 10–40 then print the strings and variables assigned in FIRST. As you can see from looking at the output, the literal string assigned in line 10 of Program 1 didn't make it. The string table points to the location where the string resided in Program 1's text, but that area of memory now contains SECOND's text. It's necessary to reassign a value to the string. However, line 15 of SECOND shows that the involving the literal string in a string operation—as in line 15 of FIRST—will create a string that can be successfully chained.

Remember, strings entered from the keyboard or from disk files will not need resetting, while those initialized with data statements will need to be reinitialized. ©

Mousor: Escape Mode Cursor For The Apple IIc

J. Blake Lambert
Tim Victor, Editorial Programmer

This short, fast utility makes it simple to use your Apple IIc mouse controller for editing in BASIC or the machine language monitor in escape mode. Disk File: MOUSOR.

Despite all the improvements Apple incorporated into the IIc, the screen editing features when using BASIC or the machine language monitor are not much better than those available on the IIe. Without an editing support package, it is difficult to copy and correct program lines. And there is no way to use the mouse controller to make editing easier.

In BASIC, usually you end up making corrections by just typing the incorrect line all over again. This wastes time and effort. The alternative is to use what is called *escape mode* editing.

"Mousor" makes using escape mode easy. By rolling the mouse over an area of the desk smaller than a 3 × 5-inch index card, you can cursor (mousor) anywhere on the screen.

How To Use Mousor

When you run Mousor, it automatically checks to see if you're using DOS 3.3 or ProDOS and then adjusts itself accordingly. When the BASIC prompt reappears, you'll have a mouse-driven escape mode cursor. If you don't understand escape mode editing, see the instructions below.

The mouse is trained to work like this:

1. Click the mouse button to activate escape mode.
2. While holding the button down, roll the mouse across the desk to move the escape mode cursor.
3. Release the mouse button to exit escape mode.

Mousor locks out keypresses while it is in escape mode, so if you want to use the escape editing functions (like ESC-E to erase the end of a line), press the ESC key.

Getting A Line Of BASIC

When you type a line of BASIC on the Apple IIc, a routine called GETLN puts the characters into a special area of memory called the input buffer. The first character on the line is stored at the start of the input buffer, and subsequent characters are added to the end of the buffer. This continues until you press the RETURN key to enter the line (with a few important exceptions). The computer clears the rest of the current screen line and stores the carriage return character into the input buffer to mark the end of the line.

When you make a mistake while entering the line, like typing the wrong character, it's easy to fix. For example, if you are entering a line such as 10 PRINT "HELLP" and notice you pressed P instead of O, you can press the left-arrow key (also called backspace) to back up and change the letter. Instead of storing the backspace in the buffer like other keypresses, GETLN treats it differently.

GETLN keeps track of the length of the input buffer by pointing to the end. When GETLN receives a backspace character, it lowers the value of the pointer by one. This removes the last character in the buffer, so all you need to do is continue typing.

If you don't notice your mistake until you've typed in several more characters, you can use the left- and right-arrow keys to make the correction. Press the backspace key until the cursor is on the letter you want to change. Type the correct letter, and then press the right-arrow key (also called retype) until the cursor returns to the end of the line.

Each time you press the retype key, the character currently under the cursor is added to the input buffer and the cursor is moved to the right. In effect, you have removed several characters from the buffer, changed the character you wanted to correct, and retrieved the rest of the characters one by one from the screen.

Now For The Tricky Stuff

Unfortunately, you can't always catch your typing errors before you press RETURN. Often, you don't even know there's a problem in a line until you've run the program. Since the retype key allows you to pick up characters from the screen and add them to the input buffer, it would be handy if you could copy most of the bad line and type only the characters you want to change. This requires a way to move the cursor around the screen without affecting the input buffer.

Pressing the ESC key puts the IIc into escape mode. In this mode,

the arrow keys move the cursor but don't change the input buffer. The IIc indicates escape mode by displaying a different cursor—an inverse plus sign. To leave escape mode, press ESC again.

Suppose you want to edit the following line in escape mode:

```
100 PRINT "THIS IS A TEDT"
```

If the line is not on the screen, you'll need to LIST 100. Press ESC to enter escape mode and move the cursor up to the 1. At this point, the input buffer is empty. Press ESC again and use the retype key to place the cursor on the D. This enters all but the last three characters into the input buffer. Now type the letter S, and press retype twice, followed by RETURN. If you like, LIST 100 to verify the correction.

To edit the same line with Mousor, you would LIST the line, click the mouse button and drag the cursor to the 1, and release the mouse button. After this, follow the same editing procedure.

Mouse Moves

You can also use escape mode to grab pieces of program lines. Mousor is especially adept at this, since movement is so easy and quick. Just keep in mind that when the mouse button is down no characters are added to the buffer.

To copy a line, first LIST it. Then enter the number for the new line you want to create, click the mouse button and drag the escape mode cursor to just beyond the original line number, and release the button. Copy the line by pressing the right-arrow key until you reach the last character, and then press RETURN.

Inserting is another useful technique. LIST the line first, then mousor (click and drag the escape mode cursor) to the beginning of the line number. Release the mouse button and right-arrow across the line until you reach the point where you want to insert characters. Press the mouse button and mousor to a blank line on the screen, then release the button and type the insert characters. Click and drag up to the listed line again, release the button, and right-arrow to the end. After making any changes, don't forget to press the RETURN key to enter them. ©

Apple ProDOS Variable Lister

Paul F. Stuever

This fast machine language utility takes the pain out of debugging BASIC programs by listing the current value of every program variable. You can also make a hardcopy of the variable list. ProDOS is required. Disk File: VAR.LIST.

How many times have you run a program, only to get a message like OVERFLOW ERROR IN 240, or worse yet, BAD SUBSCRIPT ERROR IN 240? When you list the line in question, it may look something like this:

```
240 A$(X +XZ,2 * (B / 4 = C + 1),B / 4)
    = STR$ (Z)
```

To locate the error, you'll need to type PRINT X, followed by PRINT XZ and so on, to find the current value of each variable. This is a slow, tedious way to debug a program, especially when you find that some of these variables were defined with other formulas.

"Apple Variable Lister" takes the drudgery out of such debugging tasks by quickly listing the current value of every variable in your program. The program is written in machine language and works on any Apple II series computer with ProDOS.

You can use this utility even if you don't understand machine language. Enter **BLOAD VAR.LIST** to load it into memory, followed by HIMEM: 31000 to set the top of memory. You will ordinarily want to do this at the beginning of a programming session. To list your variables, simply type CALL 32000 and press RETURN. The same command can run the routine from within a BASIC program. To make a hardcopy of the variable list, enter PR#1 before calling the routine.

A Chronological List

Variable Lister displays your program's arrays first, followed by floating point, string, and integer variables. The variables are displayed in chronological order (the order they are used in the program), not alphabetically. Although Applesoft BASIC allows arrays with up to 88 dimensions and as many elements per dimension as available memory will allow, Variable Lister is more restrictive. For this program, arrays are limited to three dimensions and a maximum of 254 elements per dimension. Attempting to list a larger array—for example, the one created by DIM A$(500)—crashes the utility.

Note that Variable Lister cannot display a variable until it has actually been used in the program. For instance, consider the following line:

```
10 A$="YES":IF A$="NO" THEN
   B$="OK"
```

Since the IF condition can never be satisfied, B$ will not appear on the variable list unless the program uses it elsewhere. This is no problem when debugging, since you're interested only in variables that were used up to the time the program crashed. However, to make a complete variable list for permanent documentation, you'll need to run your program until you know that every variable has been used. ©

Easy Apple Screen Editing

Roland Brown
Enhanced version by Tim Victor, Editorial Programmer

Here's a way to make BASIC programming easier and more fun: an advanced screen editor that makes up for the Apple's lack of full-screen editing. This outstanding programming utility works on any Apple II-series computer using either DOS 3.3 or ProDOS, in 80-column as well as 40-column mode. Disk File: EASY .APPLE.EDIT.

Although Applesoft BASIC is a powerful language, its screen editor leaves much to be desired. Some Apple II owners invest in a ROM editor, others write their programs with a word processor, and the rest just suffer with the frustrating ESCape mode editing. But ROM editors cost money, word processors don't let you flip back and forth between the text editor and BASIC to test changes, and suffering isn't *always* good for the soul. So here's a better solution: "BASIC Line Editor," a powerful utility that lets you easily modify BASIC program lines.

Start the program by typing **BRUN EASY.APPLE.EDIT** and pressing Return. The program loads at memory address $2000 (that's 8192 in decimal notation), then checks to see which operating system is present before moving itself to a safe location. This process can destroy part of a long BASIC program. If you have a long BASIC program in memory, you should save it *before* you activate the BASIC Line Editor.

Now you're ready to put the Editor to work. To edit a BASIC program line, type the ampersand (&) followed by the desired line number. For instance, enter &100 to edit line 100. The BASIC Line Editor displays the line on the screen in a format somewhat different from Applesoft's. The line is continuous rather than centered on the screen, there are no extra spaces in the line except between quotation marks, and all control characters are displayed in reverse video.

Editing Commands

The BASIC Line Editor provides 13 new editing functions. Most are accessed by pressing the Ctrl (Control) key along with a letter key. Here's a quick reference table followed by a detailed description of each command:

Key	Function
Ctrl-B	Block back
Ctrl-C	Convert hex to decimal
Ctrl-D	Delete right
Ctrl-F	Block forward
Ctrl-H	Cursor left
Ctrl-I	Insert
Ctrl-M	Return
Ctrl-S	Search
Ctrl-T	Truncate
Ctrl-U	Cursor right
Ctrl-V	Verbatim
Delete	Delete left
Esc	Return to BASIC

Ctrl-B (block back) moves the cursor back to the previous colon, or if there is no previous colon, to the beginning of the line.

Ctrl-C (convert hex) converts hexadecimal numbers to decimal. This command moves the cursor above the line being edited, prints a $ prompt on the screen, and waits for you to enter a number. This value is converted to decimal and printed. Then the cursor returns to its original position on the line.

Ctrl-D (delete right) deletes the character under the cursor. The cursor stays where it is, and everything to the right moves back one space.

Ctrl-F (block forward) moves the cursor forward to the next colon, or if there is no colon, to the end of the line.

Ctrl-H (cursor left) moves the cursor back one space.

Ctrl-I (insert) puts the BASIC Line Editor in insert mode. Any characters you type are inserted in the line until you use another Editor command.

Ctrl-M (return) is the same as pressing Return. No matter where the cursor is located on the line, pressing Ctrl-M enters the line into the program.

Ctrl-S (search) searches for the next character entered. If the search fails—in other words, if there is no character specified from the cursor to the end of the line (or if there's no such character in the line at all)—the cursor moves to the end of the line.

Ctrl-T (truncate) truncates the line at the cursor position (deletes everything after the cursor). The cursor ends up one space beyond the new end of the line.

Ctrl-U (cursor right) moves the cursor forward one space.

Ctrl-V (verbatim) lets you enter control characters verbatim. If the keypress immediately after Ctrl-V is a Control key combination, it's interpreted as a control character rather than as a BASIC Line Editor command. Ctrl-V is useful for adding Return (Ctrl-M) or backspace (Ctrl-H) characters to a line for improved printing control. If the keypress immediately following Ctrl-V is not a Control key combination, Ctrl-V has no effect. Remember that the BASIC Line Editor shows control characters in reverse video.

Delete (delete left) deletes the character to the left of the cursor and moves the cursor back one space. (The Delete key is found only on the IIe and IIc.)

Esc (return to BASIC) puts you back in BASIC. If you make a mistake when editing a line with the BASIC Line Editor, press Esc to exit back to BASIC without losing the line.

Program Notes

Activating the Editor resets the stack to the same level as does BASIC, sets up the ampersand vector ($3F5), moves the DOS buffers downward to protect DOS, and restarts BASIC. The Editor uses existing BASIC routines to read the input line and find the desired line in memory. If you try to edit a line that doesn't exist, the Editor simply returns to BASIC. If the line is found, its contents are read and listed on the screen. Text characters are listed just as they are stored. When the Editor finds a token (an encoded BASIC keyword), it locates the word in the BASIC keyword table and lists it on the screen.

Once the Editor lists the line, it enters editing mode. This part of the program gets a command from the keyboard, processes it, and updates the screen. Space doesn't permit a detailed explanation of how each Editor command works. If you're familiar with Apple machine language programming, you may find it interesting to trace through the various routines on your own.©

MultiMemory

Patrick Parrish, Programming Supervisor

This short utility partitions free memory so several BASIC programs can be loaded into your computer at once. Among other things, it's a great aid during program development—you can keep a couple of BASIC programming utilities at hand as you work, or test alternate versions of new routines before adding them to your main program. Disk File: MULTIMEMORY.

The idea of partitioning memory into several modules which can contain separate programs is not new—Charles Brannon relied on BASIC pointers to split the PET into four 8K blocks with "Quadra-Pet" (COMPUTE!, June 1981), and Feeman Ng later partitioned the 64 into three 12K blocks with "Triple 64" (COMPUTE!'s Gazette, April 1985). Much like these earlier programs, "MultiMemory" divides free memory in your Apple into four independent workspaces. You can load different BASIC programs into the computer at once. These could be utilities, applications, or games. And you can save and load programs from any of these areas without affecting the others.

But MultiMemory goes one step further. Not only are the BASIC programs in each module protected from one another, but the variables generated by each are protected as well. Any program can change a variable's value without affecting identically named variables that may exist in other partitions. That means there's even less chance of conflict between the programs, allowing you more flexibility when using MultiMemory.

MultiMemory uses the zero-page pointers at locations 103–116 to split up free memory. The program contains a short machine language routine entered by a BASIC loader.

When you run MultiMemory, line 100 sets the top of BASIC memory for the first partition. This is memory location 8192 on the Apple.

Line 110 POKEs the machine language routine in lines 150–330 into a safe place in memory. On the Apple, it area resides at location 38251 just below DOS (38400).

Line 120 checks to see if the machine language data has been correctly typed. Lines 130 and 140 place zeros in three sequential locations at the start of each memory module. The first zero is required to indicate the start of BASIC. The second and third zeros NEW each memory module. Finally, the NEW command in line 140 clears the BASIC loader from memory and leaves us in module 1.

Four Computers

To access a module, type CALL 38251 and press RETURN. Choose among work areas 1 to 4. For a test, specify number 2. A tone sounds, the partition number is displayed, and we're ready to program.

When you type LIST, you'll see that module 2 is empty. Now type PRINT FRE(0) to determine how much memory is available in this module. There should be about 8K

free—plenty of room for a short BASIC program and its variables. To see that both a program and its variables remain intact within a particular module, let's enter and run a program in module 2 and then do the same in module 1.

While in module 2, type and run this program:

```
10 REM EXAMPLE 2
20 A$="MODULE #2"
30 FOR I=1 TO 20:NEXT I
```

After running this program, type PRINT A$,I and press RETURN. You should see this:

```
MODULE #2      21
```

Save this program to disk or tape with the filename "P2."

Now let's go to another module. Before we do this, list the program in module 2 so that it's at the bottom of the screen. Now type CALL 38251 on the Apple and choose module 1.

Independent Variables

After entering module 1, type LIST to prove that our first program has been left behind in module 2. Again, if you wish, type PRINT FRE(0) to determine the amount memory available in this module. You should find approximately 6K free.

Program "P2" should still be on the screen. Even though it was listed in module 1, it remains visible if you switch partitions without clearing the screen. This makes it possible to copy program lines from one module to another with the screen editor. Simply use the screen editing keys to cursor up to line 10. Change the 2 in this line to a 1, cursor to the end of the line and press RETURN to enter this line in memory. Line 10 is now copied into memory in module 1 *without disturbing line 10 in module 2.*

If the BASIC screen prompts do not obscure the other program lines, you can copy them to module 1 in the same manner. At any rate, lines 20 and 30 should read:

```
20 A$="MODULE #1"
30 FOR I=1 TO 10:NEXT I
```

As you did before, run the program and then type PRINT A$,I. The result is:

```
MODULE #1      11
```

Save this program on disk or tape as "P1."

Now, go back to module 2 with CALL 38251 and type LIST. You should see program "P2" on the screen unchanged. Print the values for A$ and I. You'll see they still have the values they had when we left module 2.

Applications

As you can imagine, this process can be very valuable if you're writing and debugging your own programs. Suppose you're writing a program in module 1 and you need a subroutine from a BASIC program you have on disk. Maybe you aren't sure which disk the program is on.

With MultiMemory, you can nimbly jump to another module, load the directory there to find the program you need. Once you've found the subroutine you need from your earlier program, you can list it on the screen, shift back to the first partition, and copy the lines by RETURNing over them.

Now suppose that some bug in your program keeps the subroutine from working as you expected. Since variables retain their values within each program module, you can test each routine separately, compare the resulting variables, and make changes to your working version where needed.

With all this jumping from module to module, you might loose track of which partition you're in. To find out, you can type PRINT PEEK(38339).

In addition to aiding program development, MultiMemory can be used to hold a few BASIC programs that can be run individually. For instance, you might have a series of BASIC programs that, in sequence, manipulate data stored on disk. The first program could read in the data, manipulate it, and then write the results back to disk. In turn, a second or third program or even a fourth stored in the other workspaces could do the same. Or you could designate one BASIC workspace for data storage, much like a RAM disk.

Before undertaking any sophisticated programming applications with MultiMemory, however, you should consider how it works and keep in mind a few restrictions on its use.

Memory Ceilings

As mentioned earlier, MultiMemory uses a series of BASIC pointers in zero page to set up each workspace. The values required by these pointers for each module are stored in a lookup table at the end of the program. Whenever you CALL MultiMemory, it stores the current zero-page values in positions within this table which correspond to the current module. The program then waits for you to specify the next module.

When you pick a module, MultiMemory reads the zero-page pointer values for the module and places them in their proper zero page locations. The pointers transferred are the starting addresses for the location of the BASIC program, the array and nonarray variables, the string variables, and the top of BASIC memory.

The barriers separating the partitions were selected to keep MultiMemory compatible with most programs. On the Apple, the first module runs from memory locations 2048–8191; the second, from 8192–16383; the third, from 16384–27317; and the fourth, from 27318–38250.

Note that the location of module 2 coincides with the Apple's first high-resolution graphics page. That means any programs which use the hi-res graphics page should be loaded into another module. And, of course, any programs loaded into module 2 will likely be erased if a program in another module uses the hi-res page. ©

Dr. Disk

Alan H. Stein

The ability to read and edit individual bytes stored on your disk can come in handy more than once. This utility, with versions for both DOS 3.3 and ProDOS, allows you to read, change, and write to disk any block, sector, or track. Undeleting files, searching for lost passwords, and simply examining how files are stored are just some of the applications of "Dr. Disk."Disk Files: DR.DISK and DR.DISK3.3.

It's often extremely useful and even interesting to examine and change the contents of disks on a sector, block, and byte level. "Dr. Disk" is a machine language program designed to let you do just that.

Dr. Disk is easy to use. And though it's a machine language program, you don't have to know anything about machine language programming to use it. To load, run, and initialize the program (either the DOS 3.3 or ProDOS version), simply type

```
BRUN DR.DISK
```

On Your Screen

The first thing you'll see on the screen will be a request for a printer slot. If you're like most Apple II users, you have your printer connected to slot 1—simply press the 1 key. For additional flexibility, Dr. Disk accepts any key response between 1 and 7. Pressing a number key outside that range informs Dr. Disk that no printer is connected.

As soon as a key is pressed, the screen clears, except for a menu on the bottom. The menu remains on the screen except when the contents of a sector or block is being printed. This helps make using Dr. Disk a snap. Depending on which version of Dr. Disk you're using, DOS 3.3 or ProDOS, the menu should look like one of the following:

DOS 3.3

```
DRIVE 1                    READ
TRACK 11                   WRITE
SECTOR 0F                  EDIT
^S SLOT 6                  ^P PRINT
^T TOGGLE ASCII MSB ON     ^X EXIT
```

ProDOS

```
SLOT 6                     READ BLOCK
DRIVE 1                    WRITE BLOCK
BLOCK 000                  EDIT BLOCK
TOGGLE ASCII<—>HEX         ASCII MSB ON
^P PRINT BLOCK             ^X EXIT
```

Note that the first letters of drive, track, sector, read, write, and edit are in inverse.

Dr. Disk, DOS 3.3

Here are brief explanations of each of the Dr. Disk DOS 3.3 commands:

DRIVE is used to choose which drive you're accessing, 1 or 2. Obviously, this is necessary only when you have a multidrive system. When Dr. Disk starts, the default is drive 1. Press D to change it. The cursor then moves to the word *DRIVE.* Press the number of the drive you want to use.

TRACK represents, in hexadecimal, the track you want to access. A disk has 35 tracks. To access a specific track, press T. The cursor moves to the word *TRACK,* at which point you can enter the track you want. Note that you must enter it in hexadecimal. For instance, you would enter track 25 as 19.

SECTOR represents the sector you want to access. On a disk, each track contains 16 sectors. Change the sector by pressing S and the hexadecimal digit representing the sector you want. Dr. Disk defaults to track $11, sector $0F, since that sector contains the Volume Table of Contents (VTOC) for a DOS 3.3 disk.

SLOT indicates which slot your disk drive is connected to. Since most disk drives are connected to slot 6, Dr. Disk is designed to boot with that as a default. To change the slot, simply press Control-S. A cursor moves next to the word *SLOT.* Press the number of the new slot.

READ transfers the contents of a sector on the disk to memory. Simply press R, and whatever sector has been specified is read. If an error occurs, the message ERROR—PRESS ANY KEY TO CONTINUE is displayed on the bottom of the screen, and the sector reading aborts. The program continues as soon as any key is pressed.

When a sector is read from disk, the contents appear on the screen in both hexadecimal and ASCII formats. You'll see 16 lines, each containing eight bytes of information. The number of the first byte shows on the left, followed by the hexadecimal representation of the eight bytes, then finally the ASCII representation. The cursor blinks to the left of a byte and can be moved around the pattern by pressing either the arrow keys, the traditional I, J, K, M diamond (up, left, right, and down, respectively), or the Return key. Since a screen can display only half a sector at a time, when the cursor is moved past the last byte or before the first byte, the other half is automatically displayed.

WRITE writes a sector back to disk. This will generally be used

after some changes have been made. Since you can cause irreversible damage by inadvertently writing to disk, this operation is double-checked before it's allowed to proceed. The message ARE YOU SURE YOU WANT TO WRITE (Y/N)? appears at the bottom of the screen. If any key other than Y is pressed, the operation is aborted. An error during a write operation is treated the same way as an error during a read. If you try to write to an empty drive, you'll find the contents of the last sector you accessed displayed on the screen.

EDIT allows you to change the contents of a sector. To enter the edit mode, simply press E. The message PRESS ^Z TO TERMINATE EDIT MODE displays. While in edit mode, you can change the contents of the sector simply by typing. Only the display changes at this point—no physical changes are made to the disk unless and until the write command is used.

The edit mode actually has three submodes—hexadecimal, ASCII with the most significant bit on, and ASCII with the most significant bit off. These submodes are displayed on the line starting with ^T TOGGLE and can be chosen by pressing Control-T until the desired mode is displayed. Note that you must do this *before* entering the edit mode.

If you're in one of the two ASCII modes, pressing any key other than Control-Z changes the contents of one byte to reflect the ASCII code of that key. Pressing several keys in succession causes the contents of consecutive bytes to be altered. All changes are displayed immediately. The only difference between the two submodes is the way the most significant bit of each byte is treated. The only way to exit these edit modes is by pressing Control-Z.

If you're in hex mode, you can change the contents of bytes by entering the hexadecimal values. Each time you press a *pair* of hexadecimal digits, the contents of a byte is changed. Exit this mode either by pressing Control-Z or by entering an invalid digit (anything but 0–9 and A–F).

PRINT prints the contents of the sector currently on the screen. This command is activated by pressing Control-P. One word of caution: If an incorrect slot has been designated for the printer, the program will hang, or stop operation. You'll have to BRUN Dr. Disk again.

EXIT sends you back to BASIC. When you're finished with Dr. Disk, simply press Control-X. (Here's a tip that will come in handy sooner or later: If you exit from Dr. Disk accidentally, you can get back by entering the monitor—type CALL 8192 and press Return.)

Dr. Disk, ProDOS

The commands in the ProDOS version of Dr. Disk are identical in most respects. The only exceptions are the replacement of the Track and Sector commands with the Block command and a change in how Dr. Disk displays blocks after a read.

BLOCK represents, in hexadecimal, the block you want to access. Floppy disks have 280 blocks, but hard disks may have considerably more. To access a specific block, press B. The cursor moves down to the word *BLOCK*, at which point you can press, in sequence, the three digits representing the block you want. For example, to access block 002, you'd press B, 0, 0, and 2. You'll usually start at that block since it contains the disk's directory.

READ BLOCK operates just as READ does in Dr. Disk, DOS 3.3, except that the screen can display only one-fourth a block at a time. When the cursor is moved past the last byte or before the first byte, another quarter of the block is displayed automatically.

Dr. Disk is surprisingly easy to use. It refuses to accept invalid commands while letting you use the keyboard in either uppercase or lowercase. Keep in mind, however, that Dr. Disk can be dangerous since you can change the contents of a sector and perhaps inadvertently destroy a disk. You should use Dr. Disk only on disks for which you have a backup copy.

I've used Dr. Disk to learn about the structure of DOS 3.3 and ProDOS, to manipulate catalogs, and even to change machine language programs. It can be extremely useful as well as a lot of fun.

The Doctor At Work

To understand the utility of Dr. Disk, let's walk through three practical examples:

- Rediscovering a forgotten password for a *Bank Street Writer* file.
- Resurrecting a file which has been unintentionally deleted (under both ProDOS and DOS 3.3).
- Creating a disk that can simultaneously hold both DOS 3.3 and ProDOS files.

Rediscovering a forgotten password is easiest, so let's consider that first even though not everyone with an Apple uses *Bank Street Writer*. The process may be different for other programs which require a password, but the ideas here should be similar.

Rediscovering A Forgotten Password

To make the explanation as simple and unambiguous as possible, it's assumed that you have a freshly initialized DOS 3.3 disk. (You can initialize a disk either through the utilities contained in *Bank Street Writer* or by typing INIT HELLO after booting any standard DOS 3.3 disk.)

Use *Bank Street Writer* to save a short text file. Call the file COMPUTE and give it the password TEST. Pretend that you forgot the password. *Bank Street Writer* makes it impossible to retrieve a file saved with a password unless you use that password. If you try to look at a catalog from within *Bank Street Writer* (or even outside *Bank Street Writer*), all you'll see is the word *COMPUTE*. Fortunately, there's a way to retrieve that password—Dr. Disk (DOS 3.3 version, obviously).

Run Dr. Disk. Conveniently, the first track/sector combination listed when you run the program is track $11, sector $0F, which happens to be precisely where the start of the disk directory is stored on a standard DOS 3.3 disk. Make sure the disk with the *Bank Street Writer* file is in the drive. All you have to do is press R, for READ, to read the contents of the sector from the disk. Part way down the right-hand column the word *COMPUTE* will stand out. Right next to it will be the word *TEST*, your forgotten password. You can now access the file again.

Why was it so easy to find the password? DOS 3.3 stores the names, locations, and other information about its files in a directory which begins on track $11, sector $0F, and continues on track $11 through sectors $0E, $0D, $0C, and so on. You can easily pick out the names of files by looking through those sectors. If the COMPUTE file had not been in sector $0F, you could just have looked further.

Why did you see the password TEST when you looked through the sector using Dr. Disk, but not when you asked for a catalog? The answer can be found by looking at the hexadecimal codes for the characters in the words *COMPUTE* and *TEST*. The codes are different, even for the same letters. That's because the characters in the password are stored as control characters rather than ordinary characters. For example, the *S* in the password TEST is stored as Control-S, hexadecimal code $93, rather than as S, hexadecimal code $D3. Since control characters are not displayed when a disk is cataloged, the password is invisible. It's actually part of the file's name, however, and is easily seen when examining the directory sector with Dr. Disk.

Undeleting A File Under DOS 3.3.

To see how to undelete a file using Dr. Disk, DOS 3.3, we'll assume you have a newly initialized DOS 3.3 disk with a single file on it—an Applesoft program named HELLO which prints out THIS IS A TEST FOR COMPUTE! when executed.

Use Dr. Disk to examine track $11, sector $0F, of the sample disk. If you look at the ASCII characters on the right, you'll see the word *HELLO* starting at byte $0E. If you look at the hexadecimal codes for the three bytes ($0B, $0C, and $0D) preceding the beginning of the filename, you'll see *12 0F 02*. These give the track ($12) and sector ($0F) of the "track/sector list" for the file along with a code ($02) for the file type (Applesoft). After a series of $A0's (the code for a blank), you'll see a *02*, which stands for the length of the file in sectors.

Exit from Dr. Disk by pressing Control-X and look at the disk's catalog by typing CATALOG. You should see one file listed, the HELLO program. Delete that file by typing DELETE HELLO. If you look at the catalog again, no files will show. Now go back to Dr. Disk. (This can be done in two ways—Dr. Disk can be BRUN again, or you can type CALL 8192 and hit Return. The latter method will work as long as nothing has been done to disturb the image of Dr. Disk in memory.)

If you now examine the same sector of the sample disk, you'll see exactly two changes. Byte $0B has been changed from 12 to FF to signify a deleted file. And byte $2B has been changed from 00 to 12. (This byte comes just before the byte containing the file length.) All you have to do to get HELLO back into the catalog is change those two bytes back to their original states (easy) and update the VTOC (Volume Table of Contents—not so easy).

To change the two bytes in this sector, first move the cursor to byte $0E by using either the arrow keys or the I-J-K-M key diamond. Next, perform the following steps:

- Press Control-T to get to hex mode for input.
- Press E to enter edit mode.
- Press 1 and then 2 to change byte $0B to 12.
- Press Control-Z to get out of the edit mode.
- Move the cursor to byte $2B.
- Press E to reenter edit mode.
- Press A and then 0 to change byte $2B to A0.
- Press Control-Z to get out of the edit mode.
- Press W to write the sector to disk.
- Press Y to confirm that you want to write to disk.

You've just put HELLO back into the catalog. You could now exit from Dr. Disk and see HELLO in the catalog and even execute it; however, that would be living dangerously. Here's why: DOS 3.3 uses a special table called the VTOC (Volume Table of Contents) free sector bitmap to determine which sectors are in use. The VTOC has not yet been updated. If you don't update the VTOC and then try to save another file, the first file may be overwritten, creating garbage. There are two ways to update the VTOC. One is routine, but very slow; the other is much faster, but also much more involved.

The routine but slow method consists of taking an initialized disk and using *FID* or a similar utility to copy every file of your original disk to the new disk. *FID* will automatically take care of the VTOC, but every file must be transferred individually, and the original disk must be reinitialized.

The quicker way is to use Dr. Disk again. To use this method to undelete any file, you need some understanding of how DOS 3.3 stores the directory. Also, if you want to reconstruct the VTOC manually rather than by recopying each file to a separate disk, details of the VTOC are necessary. The following explanation should suffice. Apple's reference manuals and the excellent reference *Beneath Apple DOS*, by Don Worth and Pieter Lechner, can supply more detailed information.

Undeleting A File Under ProDOS

The process of undeleting a file under ProDOS bears some similarities to the corresponding process under DOS 3.3. Let's go through another simple example.

Again, you should have a formatted ProDOS disk, with the volume name /COMPUTE, containing a single short Applesoft program named TEST. This may be created by formatting a disk using the ProDOS filer and then saving a short, or even blank, program under the name TEST.

Execute Dr. Disk (ProDOS version) and put in the sample disk. Look at block $002 by pressing B 002 R (B for block, 002 for block $02, R for read block). The relevant parts are bytes $05–$13, containing the volume directory name COMPUTE; byte $25, containing $01, representing the fact that the /COMPUTE directory contains one file (TEST); bytes $2C–$3A, containing the filename TEST; byte $2B, containing $14, which shows that this is a *seedling* file with a filename $04 bytes long; and byte $3C, containing $07, which shows that the file's *key block* is block $007.

Exit Dr. Disk by pressing Control-X. You can type CAT to see the

How DOS 3.3 Stores The Directory

DOS 3.3 stores its directory, as mentioned earlier, starting on track $11, sector $0F. On each directory sector, bytes $01 and $02 (the second and third bytes, since the bytes are numbered from $00) contain a *link* to the next directory sector. Byte $01 contains the track, while byte $02 contains the sector. The actual directory entries begin with byte $0B. Each entry contains $23 (35 in decimal, but it's more convenient to think in hexadecimal) bytes. Let's number these bytes $00–$22, remembering that this is relative to the start of the entry rather than the start of the sector.

Byte $00 of the entry contains the track of the *track/sector list*, while byte $01 contains the sector of that list. Byte $02 contains a code for the file type. The name of the file begins at byte $03 and continues through byte $20, a total of 30 (decimal) bytes, which is the maximum length of a filename. It's generally easiest to recognize the start of the filename and work from there to find the other fields. Finally, bytes $21–$22 contain the length of the file. The only bytes that concern us are bytes $00, $01, and $20. Indeed, the only bytes changed in the directory when a file is deleted are bytes $00 and $20. The contents of byte $00 is transferred to byte $20 and replaced by $FF. DOS 3.3 recognizes a deleted file simply by finding $FF at byte $00. All you have to do to the directory entry to undelete the file is find the contents of byte $20 and copy it to byte $00. Since byte $20 actually contains the last character of the filename, which usually is a blank, it's good practice to change byte $20 to $A0, the code for a blank, as well.

Updating the VTOC is considerably more complicated, so you may want to skip the next section and update the VTOC by copying each file to a fresh disk. (An alternative and perhaps faster method would be to copy the undeleted file to a separate disk, delete it again (!) from the original disk, and then recopy it from the separate disk to the original disk.)

The VTOC resides on track $11, sector $00, and contains a bitmap of free sectors. The status of each track is contained in a group of four bytes. These groups of bytes start at byte $38 and continue through byte $C3. Bytes $38–$3B contain the status of track $00, bytes $3C–$3F contain the status of track $01, bytes $40–$43 contain the status of track $02, and so on, continuing to bytes $C0–$C3, which contain the status of track $22.

Within each group of four bytes, only the first two are used—the last two are set to $00. The first two bytes contain a total of $10 (16 in decimal) bits, one for each sector on the track. Each bit set to 1 corresponds to a sector which is free, while each bit set to 0 corresponds to a sector which is in use. The bits of the second byte, from low bit to high bit, represent sectors $0–$7, while the bits of the first byte, also from low bit to high bit, represent sectors $8–$F. This may seem backward, but the reason should become clear from the following example.

Suppose sectors $02, $07, $0C, and $0F are free on track $03, while all other sectors of track $03 are in use. Write down 16 ($10) binary digits next to each other, numbering them from $00 to $0F, right to left.

```
0F 0E 0D 0C 0B 0A 09 08 07 06 05 04 03
   02 10 00
```

Set digits $02, $07, $0C, and $0F to 1, and all the others to 0. You should get the following binary number:

```
0F  0E  0D  0C  0B  0A  09  08  07  06  05  04  03  02  10  00
1   0   0   1   0   0   0   0   1   0   0   0   0   1   0   0
```

Now convert that binary number to hexadecimal. You should get $9084. In the VTOC free sector bitmap, bytes $44–$47 correspond to track $03. Byte $44 would contain $90, byte $45 would contain $84, while bytes $46 and $47 would each contain $00.

In order to update the VTOC when undeleting a file, you must determine which sectors are used by the file and clear the corresponding bits in the free sector bitmap to 0.

To determine which sectors are used by a given file, refer to its track/sector list. As described earlier, the location of the first sector in the track/sector list, is pointed to by bytes $00–$01 of the file's directory entry. The following is a brief description of the relevant portions of each sector of the track/sector list.

Bytes $01 and $02 (the second and third bytes, remember) point to the next sector in the track/sector list. The last sector of the track/sector list contains $00 in these bytes. Then, starting with byte $0C, each pair of bytes gives the track and sector of a sector used in the file. Bytes not needed to give the locations of sectors in use contain $00.

Thus, you have the following process for updating the VTOC. First, you must look through the track/sector list and make a list of all sectors used by the file. (Don't forget that the sectors containing the track/sector list are themselves in use and must be included in the list.) You must then clear each corresponding bit in the free sector bitmap to 0. This is a tedious process, prone to error. On occasion, I've found it fun to do, but it's generally much safer to recopy the files and let *FID* do the update.

catalog, which will contain the single file TEST. (Recall that ProDOS commands can be entered from the monitor, with the * prompt, and that you don't have to reenter BASIC.) Now delete the file by typing DELETE TEST. You can check that TEST is no longer in the catalog. Reenter Dr. Disk by typing 2000G (from the monitor) and read in block $002 again.

You'll see the following changes: Byte $25, giving the file count, has been changed to $00 since there are no longer any files in the directory, while byte $2B, giving the storage type and name length, has been changed to $00 to

indicate that the file TEST has been deleted.

You can undelete TEST by moving the cursor to byte $25 and pressing T, E, 0, and 1, then Control-Z to change byte $25 back to $01. Change byte $2B back to $14 in the same way. You must, of course, remember to write the changes back to disk by pressing W and then Y.

You also have to update the Volume Bit Map, the ProDOS counterpart of DOS 3.3's VTOC free sector bitmap. As with DOS 3.3, this can be done in two ways. You can use the ProDOS *Filer* program the same way you used the DOS 3.3 *FID* program and let *Filer* figure out what to do with the Volume Bit Map, or you can update the bitmap directly. In this particular case, you can update the bitmap by changing byte $00 of block $006 to $00. (In general, since updating the ProDOS Volume Bit Map is a more convoluted process than updating the DOS 3.3 VTOC free sector bitmap, I advise using *Filer*.)

As with DOS 3.3, to undelete arbitrary files, you must know a little more about the structure of ProDOS. You can find more detailed information in Apple's *ProDOS Technical Manual* or in *Beneath Apple ProDOS*.

Creating A Disk Which Can Contain Both ProDOS And DOS 3.3 Files

This process of creating a disk which can store both ProDOS and DOS 3.3 files is simpler than you might think. This is because ProDOS and DOS 3.3 keep their vital information in different places. Furthermore, once you create one such disk, you never have to go through the process again, since you can use any standard copying program, such as *COPYA* or the ProDOS Copy command in *Filer*, to duplicate your prototype.

(Note: The process described here is not original. I've found it in several places, including CompuServe. Naturally, I didn't learn that until I had gone through the painful process of figuring it out myself.)

To create your double-DOS disk, follow these steps:

1. Format a ProDOS disk.
2. Initialize a DOS 3.3 disk and delete the HELLO program.
3. Copy track $11 from the DOS 3.3 disk to the ProDOS disk. This is the track that contains the VTOC and catalog sectors. This can be done in any one of the following three ways.

Using Dr. Disk, DOS 3.3 version. Successively copy each sector, $00–$0F, from the DOS 3.3 disk to the ProDOS disk as follows.

- Boot Dr. Disk and press T, 1, and 1.
- Put the DOS 3.3 disk in your drive. Press S, 0, and R (this will read sector 0).
- Take out the DOS 3.3 disk and put in the ProDOS disk.
- Press W and Y (this will write sector 0 to the ProDOS disk).

Repeat this process for sectors $01, $02, and so on, to sector $0F.

This can be speeded up if you have two drives, in which case rather than swapping disks you can alternately specify which drive you want to read from or write to.

Using Dr. Disk, ProDOS version. Successively copy blocks

ProDOS File Structure

Starting with block $002 and usually continuing on consecutive blocks, ProDOS contains *directory headers* and *file descriptive entries*. Bytes $00–$03 of each block contain links to similar blocks. The important parts of a directory header, at least for our purposes, are the *name* and the *file count*. The name is important because you'll usually be able to recognize the name of a directory and from that name locate the file count, which occurs $20 (32 decimal) bytes after the start of the directory name. Thus, when you undelete a file from a given directory, you must first find the directory name and then look $20 bytes further on. Conveniently, this is exactly four rows below the first character of the filename on the Dr. Disk display. (One complication: We recognize the filename from its ASCII characters, which may flash or display in inverse on our display, while we manipulate the hexadecimal codes for the file count.) Just increase the byte containing the file count by $01. (One more complication: If the byte contains $FF, change it to $00 and then increase the next byte by $01. This is because the file count is actually stored in two bytes. You'll rarely have to be concerned with the upper byte, however.)

The relevant parts of the file descriptive entry are its *name*, *storage type*, and *name length*. The name is easy to find and doesn't change. The byte before the name contains both the storage type and name length. That byte changes to $00 when a file is deleted and must be changed back when the file is undeleted.

The contents of the byte may be split into two hexadecimal digits. The second digit represents the length of the filename. That can easily be determined by looking at the filename and counting up its characters, remembering of course to count in hexadecimal. The first digit represents the storage type. Generally, that digit will be either $1, $2, or $3, corresponding to a *seedling*, *sapling*, or *tree file*. These files represent Apple trees in various stages of growth. Actually, a seedling file is a file containing just 1 data block, a sapling is a file containing from 2 to 256 data blocks, and a tree is a file containing from 257 to 32,768 data blocks. To resurrect a deleted file, therefore, you must remember, roughly, a file's length. (Most files will probably be saplings, in the range of 2 to 256 data blocks.)

In our example, the file TEST used just one block; hence, it was a seedling file and the first digit was 1. The length of the name TEST was 4, so the second digit was 4. The byte immediately before the name, then, should be changed to $14.

If you remember the approximate size of a deleted file, the ProDOS undeletion process is a simple two-step process: Increase the file count of the directory header by $01 and replace the byte containing the storage type and the name length in the file descriptive entry. Of course, don't forget to use FILER to tediously update the Volume Bit Map.

$088 to $08F from the DOS 3.3 disk to the ProDOS disk. These are the blocks, according to ProDOS, which correspond to DOS 3.3 track $11. Follow these directions:

- Boot Dr. Disk, ProDOS version.
- Put the DOS 3.3 disk in your drive.
- Press B, 0, 8, 8, and R (this reads block $088).
- Replace the DOS 3.3 disk with the ProDOS disk.
- Press W and Y (this writes block $088 to the ProDOS disk).

Repeat the process for blocks $089, $08A, and so on, until block $08F. As with the process using the DOS 3.3 version of Dr. Disk, you can do this without disk swapping if you have two drives. Clearly, it's faster to copy the track using the ProDOS version of Dr. Disk rather than the DOS 3.3 version.

Copy utility. The quickest method of all is to use a program which has the capability of copying individual tracks. (But you may not have such a utility, in which case Dr. Disk will be invaluable.)

4. Discard the DOS 3.3 disk. Everything from here on refers to the disk that originally contained only ProDOS.
5. Change the Volume Bit Map to make ProDOS think that everything from block $088 on is used. This can be done using either version of Dr. Disk. If you're using the ProDOS version, you'll be dealing with block $006 (press B, 0, 0, 6, and R); if you're using DOS 3.3, you'll be dealing with track $00, sector $03 (press T, 0, 0, S, 3, and R).

   With either Dr. Disk, change the contents of bytes $11–$22 to 00's. Make sure you're in hex mode before you enter the editing mode. Press W and Y when you're finished to write the changes to disk.
6. Change the DOS 3.3 VTOC free sector bitmap to make DOS 3.3 think that every sector in tracks $00–$10 is used. If you're using the ProDOS version of Dr. Disk, you'll be dealing with track $11, sector $00.

   With either version, change the contents of bytes $44–$7C to 00's and make sure you write the changes to disk.

You now have a disk that can hold both types of files. If you're using ProDOS, files will be saved only to blocks $000–$087, while if you're using DOS 3.3, files will be saved only to tracks $11–$22. In either case, you have fooled the operating system into thinking the rest of the disk is already written to, which it very well may be. If you analyze what we've done more carefully, you'll realize that the way we tampered with the Volume Bit Map and the VTOC was somewhat arbitrary. Different portions of the disk could be allocated to each system.

As it stands, the disk is unbootable. However, due to the nature of ProDOS, it's a simple matter to make it boot ProDOS while still retaining its dual-DOS capacity. All you need to do is copy two files to it: PRODOS and BASIC.SYSTEM. The disk would then act like an ordinary bootable ProDOS disk if booted, but it would have a limited storage capacity since about half the disk was marked off-limits. But it would retain the ability to store DOS 3.3 files. ©

Compress And Decompress

Jason Coleman

This pair of Apple II programs helps you store more hi-res picture files on a disk. ProDOS is required. Disk Files: COMPRESS, DECOMPRESS.

"Compress" and "Decompress" are two programs designed for use with hi-res graphics files for Apple II series computers using ProDOS. Their purposes are simple. Compress, packs a hi-res picture file into a smaller space than normal. Decompress unpacks the picture.

Compress displays instructions for using both programs, and it prompts you through every step of the process of compressing a picture file. Simply follow the on-screen instructions.

Decompress reverses the process. Use the following command to install the program:

BRUN DECOMPRESS

Next, activate the high-resolution graphics area with the following command:

HGR

Now whenever you want to load a compressed picture file, use a command of the following form:

DECOMPRESS ***filename, slot, drive, address***

You must always supply a filename to tell this program which file to decompress. The remaining parameters are optional and need only be supplied if you wish to use a different slot, drive number, or load address. You may specify the load address in either decimal or hexadecimal (base 16) numbering. To specify a hexadecimal load address, put a dollar sign ($) in front of the address. ©

The information and program-packed Apple magazine now available six times a year!

COMPUTE!'s Apple APPLICATIONS

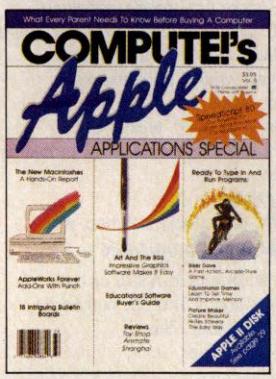
A bimonthly magazine for every Apple II and Macintosh owner. Includes informative features and top-quality programs ready to type in and run.

Millions of Apple personal computers—from the venerable II+ and IIe to the compact IIc and the powerful IIGS—are in homes, schools, and offices across the country. And Apple's Macintosh computers are showing up in more businesses, more universities, and now more homes than ever before. Apple computers are being used for almost everything—entertaining and educating people of all ages, managing home finances, running businesses, writing, painting, composing. Name something computers can do, and Apple computers do it.

That's why COMPUTE! Publications created COMPUTE!'s Apple Applications magazine as a semiannual nearly three years ago. Its informative features, product reviews, tutorials, and type-in programs have made the magazine extremely successful on the newsstands. So successful, in fact, that we're making COMPUTE!'s Apple Applications a bimonthly magazine and offering subscriptions for the very first time.

Whether you use an Apple computer at school to teach or to learn, at home to help organize your household, or at work to keep your business profitable, you'll find valuable applications, information, and news in every issue of COMPUTE!'s Apple Applications:

- **Fascinating features.** Thoroughly researched, well-written feature articles on such wide-ranging topics as desktop publishing, computer-generated arts and crafts, and new frontiers in educational computing.
- **Top-quality programs.** Sophisticated applications such as *SpeedScript 80*, an 80-column word processor for all Apple II computers. Exciting arcade-style games such as "Heat Seeker." Colorful graphics programs such as "Picture Maker." Utilities. Educational software. Home and business productivity applications. Every program is ready to type in and run. And with COMPUTE!'s program-entry software—"Apple Automatic Proofreader" and "Apple MLX"—typing is a snap. Or, if you prefer, a companion 5¼-inch disk containing all of the programs is available singly or by subscription.
- **Hands-on tutorials.** Tips and techniques for all kinds of computer applications—from power programs like *Microsoft Word* to popular software like *AppleWorks*.
- **Apple news.** The latest information on recent Apple releases, from the introduction of the Apple IIGS to an inside look at the powerful Macintosh II.
- **Balanced reviews.** Honest and up-front evaluations of the most interesting software and hardware for the Apple II and Macintosh.
- **Buyer's guides.** Detailed reference guides to new Apple II and Macintosh software and hardware.
- **And more.** Interviews with Apple newsmakers. Comprehensive lists of Apple user groups. Forecasts of the future of computing with Apple computers. Reports from the latest industry trade shows. And in 1988, COMPUTE!'s Apple Applications will begin publishing the winning programs from our $10,000 Apple programming contest!



Look for the December 1987 issue of the new bimonthly COMPUTE!'s Apple Applications available November 1 at your favorite newsstand. Or you can have it delivered to your mailbox six times a year for only $13.95. Each issue's companion disk can be purchased separately for $12.95, or you can take a year's subscription to the disk for just $49.95—you save 35 percent over the single-disk purchase price.

As a special bonus, if you order a prepaid subscription before November 1, you'll get the first issue absolutely free! That's seven issues for $13.95!

To order a subscription to the magazine and/or disk, call TOLL FREE, 1-800-727-6937. Or send a check or money order to:

COMPUTE!'s Apple Applications
P.O. Box 10767
Des Moines, IA 50340

To order a single copy of the magazine, or to order an individual disk for $14.95 (includes $2.00 shipping and handling), please call 1-800-346-6767. (In New York, 212-887-8525.)

COMPUTE! Publications, Inc. abc
Part of ABC Consumer Magazines, Inc.
One of the ABC Publishing Companies

SAVE! Special Sale

TRIPP LITE **SURGE SUPPRESSORS**

6-OUTLET Spike BAR
6-ft. cord
ONLY $35

6-OUTLET Spike BLOCK
Direct wall plug-in
ONLY $25

Complete high voltage spike and noise protection! Prevents computer damage and errors by stopping spikes up to 140 joules at current levels to 13,000 amps. Pilot light indicates suppression components are on. Spike Bar has lighted ON/OFF switch and 15 amp circuit breaker.

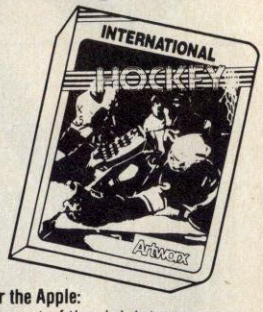
Call toll free to order....1-800-662-5021
IN ILLINOIS, CALL 1-312-648-2191

INDUS-TOOL, 730 W. Lake St., Dept. BOA, Chicago, IL 60606

Enclosed is $_____ for ☐ $25 Block ☐ $35 Bar
Charge on my ☐ MasterCard or ☐ Visa *Expires*_____
Card No. _____

Name _____
Company _____
Address _____
City, State, Zip _____
Phone _____

Score the winning goal with International Hockey for the Apple.



Examine Artworx' unique line of software for the Apple:
INTERNATIONAL HOCKEY brings the excitement of the rink into your own home. Complete with passing, fights and penalty shots! **Only $24.95**
STRIP POKER is *the* classic computer program. Play against Suzi and Melissa; the more you win, the more they take off! **Only $29.95**
LINKWORD LANGUAGES let you learn a foreign language *fast!* Available for Spanish, French, German, Italian, Russian, Dutch, Greek and Portuguese. All languages come with audio tape. **Only $29.95** each.
BRIDGE 5.0—our popular bridge playing program now has improved bidding and play plus many new features and refinements including auto-play and being able to request an opening hand. **Only $29.95**
COMPUBRIDGE is our bridge tutorial. Learn to play bridge the easy way with 10 text chapters and 8 random quizzes. **Only $24.95**
MONKEYMATH and MONKEYNEWS teach math facts and reading comprehension in a most enjoyable way. Ages 5-10. **Only $17.95** each.

Artworx®

Artworx Software Co., Inc., 1844 Penfield Road, Penfield, N.Y. 14526 (716) 385-6120 · (800) 828-6573

Public Domain & User Supported Software

NEW TOP TEN FOR APPLE $5.00/DISK

☐ 037 FREEWRITER wordprocessor (Apple II + needs paddles)
☐ 038 BUSINESS/HOME MANAGEMENT checkbook, calculator, more
☐ 039 BEST OF BUSINESS general ledger, payroll, much more
☐ 056 BANK'n SYSTEM check balancer, write & print checks
☐ 057 OMNI FILE data base with instructions
☐ 064 BEST OF EDUCATION math drills, spelling, typing, etc.
☐ 085 BASIC MATH DRILLS fractions, multiple choice, work problems
☐ 118 GAMES fast action space arcade games
☐ 195 PASSTIME, a potpourri of programs
☐ 213 BEST UTILITIES diskcat, krunch, diskcheck, diskmap, etc.

NEW TOP TEN FOR MAC $9.00/DISK

☐ 005 CODE CRACKING, FEDIT edit file blocks in ASCII or hex
☐ 006 ResED and ReED edit menu bars, icons and I.D. numbers
☐ 007 SWITCHER edit multiple Microsoft BASIC files
☐ 029 COMMUNICATIONS Red Ryder, MacTep
☐ 037 SLIDE SHOW
☐ 039 FONTS Font catalog
☐ 045 DESK ACCESSORIES Minifinder, timer
☐ 062 GAMES Dungeons of doom, baseball
☐ 067 GAMES Billiards, volleyball, juggling
☐ 086 BEST OF MAC MacWorld 86

PUBLIC DOMAIN SOFTWARE EXCHANGE
Authorized Dealer

Add $4 shipping & handling per order.
CA residents add 6.5% sales tax

Call toll free **800-431-6249**
in Calif. **415-952-1994**

Amount enclosed $ _____ ☐ Check
☐ VISA ☐ MasterCard

Card No. _____
Signature _____ Exp. Date _____
Phone ( ) _____
Name _____
Address _____
City _____ State _____ Zip _____

BLACKSHIP
COMPUTER SUPPLY
P.O. Box 883362
San Francisco, CA 94188

PRINTER MANIA!!!

CALL 1-800-288-8088

APPLE IIC PARALELL INTERFACE
JUST PLUG & PRINT!
NOW ONLY $79.95!

PANASONIC

1080i	$155.00
1091i	$175.00
1092i	CALL
1592	CALL
1595	$419.00
3131	CALL
3151	$399.00

EPSON

FX86E	CALL
FX286E	$445.00
EX800	$395.00
EX1000	$540.00
LQ800	$375.00
LQ850	$490.00
LQ1000	$499.00
LQ1050	$660.00

CALL FOR OTHER SOFTWARE AND HARDWARE

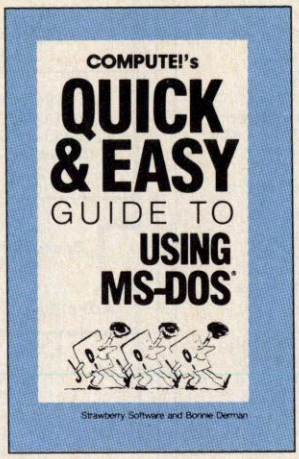
10 DISKS $10.00!

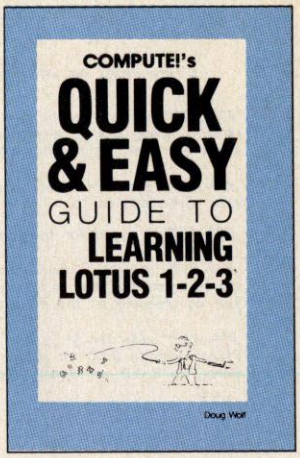
CALL FOR PRICES ON STAR PRINTERS, H.P. PRINTERS TOSHIBA PRINTERS MODEMS AND MONITORS

MICRO WORLD ELECTRONIX, INC.
SALES 1-800-288-8088 CUST. SVC. (303) 988-5907
Manufacturer's warranty honored. prices subject to change.

COMPUTE! Books introduces a new line of easy-to-use computer books.

COMPUTE!'s Quick and Easy Guides are effective, dependable reference guides to using applications on your personal computer. From command summaries and quick-reference charts to tips for power users, you'll have the information you need right at your fingertips. Each book is specially designed to be convenient and easy to understand.





COMPUTE!'s Quick and Easy Guide to Using MS-DOS
Bonnie Derman and Strawberry Software
ISBN 0-87455-105-6 $12.95
Whether you're an experienced MS-DOS user or just starting out, this book is *your* easy-to-use reference to this powerful and flexible operating system for IBM and compatible computers. No longer will you need to wade through hundreds of pages to find just the right DOS command. *COMPUTE!'s Quick and Easy Guide to Using MS-DOS* puts the information right at your fingertips. With two indices and an expanded table of contents, simply look up the task you want to accomplish or the DOS command you need—and you'll find an explanation of exactly what to do.

COMPUTE!'s Quick and Easy Guide to Learning *Lotus 1-2-3*
Doug Wolf
ISBN 0-87455-106-4 $12.95
The perfect guide for the novice user of the popular and powerful *Lotus 1-2-3*, this book assumes no previous computer experience whatsoever. *COMPUTE!'s Quick and Easy Guide to Learning* Lotus 1-2-3 covers everything from building a spreadsheet model, setting up a database, and using *1-2-3* for word processing, to creating graphs and powerful macros. Also included are discussions on how to start *1-2-3* and available add-on software that can be used with *Lotus 1-2-3*.

COMPUTE!'s Quick and Easy Guide to *AppleWorks*
Tom and Ellen Dougher
ISBN 0-87455-109-9 $10.95
AppleWorks, the unchallenged leader in personal productivity software on the Apple II series, is an integrated package combining a word processor, database, and spreadsheet. It's a complete program that lets you do everything from writing letters and reports to keeping extensive files and projecting financial earnings. With *COMPUTE!'s Quick and Easy Guide to* Appleworks, you'll see how to get started with *AppleWorks*, and how to use each of the three major applications. A Command Summary, Keyboard Template, and Quick-Reference Chart provide information in one-stop locations. Numerous ideas offer practical examples on new ways to use *AppleWorks*. And there's even all the latest information on *AppleWorks*' add-ons and enhancements. Clearly defined sections apply to novices and power users alike.

COMPUTE!'s Quick and Easy Guide to *dBase III Plus*
Chuck Doherty
ISBN 0-87455-107-2 $10.95
For the beginning or experienced user of this popular database management program. Ashton-Tate has sold more than a million copies of *dBase*, and *COMPUTE!'s Quick and Easy Guide to* dBase III Plus is *the* guide and reference to help you tap all the power and flexibility that *dBase III Plus* has to offer. This is not a manual for *dBase*; rather, it's a guide that makes getting started with *dBase* easy and finding specific information quick. For all users of *dBase III Plus* on the IBM PC, PC XT, PC AT, or compatible computer.

COMPUTE!'s Quick and Easy Guide to *WordPerfect*
Patricia Farrell
ISBN 0-87455-011-4 $10.95
Write, edit, format, and print with *WordPerfect* your first day, using this simple-to-use guide to the most popular word processor for the IBM and IBM compatibles. Organized so that you can immediately begin to use this impressive software, this book covers all the basics and many of the program's advanced features. Read what you need and then start to work with *WordPerfect*. Learn how to enter, edit, shift, and print text. Use the more advanced techniques when you're ready. See how to prepare a complete report, set text columns, merge information to create form letters, and create special forms.

COMPUTE!'s Quick and Easy Guide to Desktop Publishing
Dan McNeill
ISBN 0-87455-112-9 $10.95
With a computer, a printer, and the right software, you can turn out documents, flyers, brochures, and more—all of the highest quality available outside a professional print shop. This easy-to-use reference guide explains the ins and outs of desktop publishing, shows what you can do with the variety of available software, and even helps you select your system. If you're tired of wading through long and technical manuals, this guide and tutorial will get you started today with desktop publishing.

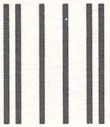
COMPUTE!'s Quick and Easy Guide to Dow Jones News/Retrieval
Donald B. Trivette
ISBN 0-87455-113-7 $10.95
Learn how to more efficiently use the Dow Jones Information Service, a powerful business and stock information database. Access up-to-date stock quotations, financial information, company background data, and more—all with Dow Jones. Using a personal computer, a modem, the phone, and the right software, you can search through back issues of publications like the *Wall Street Journal* or directly place orders. This simplified guide to the preeminent business information service puts everything within easy reach, and gets you started faster and more easily than any manual.

Look for COMPUTE!'s Quick and Easy Guides at your local book or computer store.
To order any COMPUTE! book, call toll free **1-800-346-6767** (in NY call 212-887-8525)
or write COMPUTE! Books, Customer Service, P.O. Box 5038, F.D.R. Station, New York, NY 10150.
Customer Service hours are 10:00 a.m.–12:30 p.m. and 1:30 p.m.–3:00 p.m. EST Mon. through Fri.

COMPUTE!™ Publications, Inc. abc
Part of ABC Consumer Magazines, Inc.
One of the ABC Publishing Companies

COMPUTE! books are available outside the United States from subsidiaries of McGraw-Hill International Book Company.

Advertisers Index



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 7478 DES MOINES, IA

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE!'s
Apple

P.O. Box 10955
Des Moines, IA 50347-0955

SUBSCRIBE TO APPLE APPLICATIONS!

☐ $13.95 One Year US Subscription
☐ $26.95 Two Year US Subscription

Name _____

Address _____

City _____ State _____ Zip _____

☐ Payment Enclosed ☐ Bill me
Charge my: ☐ VISA ☐ MasterCard American Express (Call 1-800-247-5470)
Account No. _____ Expires ___/___

Your subscription will begin with the next available issue. Please allow 4–6 weeks for delivery of first issue. Subscription prices subject to change at any time. Outside the U.S.A. please add $6 for each subscription.

JNB89

For Fastest Service,
Call Our **Toll-Free**
US Order Line
1-800-727-6937

Reader Information Service

Use this card to request FREE information about products advertised or reviewed in this issue. Circle the numbers that correspond to the key number appearing in the advertisers index. New Products numbers are found within the issue. *COMPUTE!'s Best of Apple* cannot be responsible if advertisers do not provide literature to readers.

Please use this card *only* for requesting product information.

101	102	103	104	105	106	107	108	109
110	111	112	113	114	115	116	117	118
119	120	121	122	123	124	125	126	127
128	129	130	131	132	133	134	135	136
137	138	139	140	141	142	143	144	145
146	147	148	149	150	151	152	153	154
155	156	157	158	159	160	161	162	163
164	165	166	167	168	169	170	171	172
173	174	175	176	177	178	179	180	181
182	183	184	185	186	187	188	189	190
191	192	193	194	195	196	197	198	270

Please print or type name and address clearly. Limit one card per person.

Name _____

Address _____

City _____

State/Province _____ Zip _____

Country _____

Please include zip code. Expiration 1/31/88

Please answer the following questions to help us fill your computing needs.

1. Please let us know. Do you

	Own:	Plan to buy:
Apple II+	☐ 270	☐ 271
Apple IIc	☐ 272	☐ 273
Apple IIe	☐ 274	☐ 275
Apple IIcs	☐ 276	☐ 277
Apple III	☐ 278	☐ 279
Macintosh 512	☐ 280	☐ 281
Macintosh Plus	☐ 282	☐ 283
Macintosh SE	☐ 284	☐ 285
Other _____ (please specify)	☐ 286	☐ 287

2. Where do you usually use your Apple computer?
At home/home office ☐ 214
At a business office ☐ 215
Both office and home ☐ 216
In the classroom ☐ 217
Other _____ 218

3. Do you plan to purchase a personal computer in the next 12 months?
Yes ☐ 219
No ☐ 220
Don't know ☐ 221

4. Do you plan to purchase software in the next 12 months?
Yes ☐ 222
No ☐ 223
Don't know ☐ 224

5. Which of the following peripherals do you own or use? (check all that apply)
Disk Drive ☐ 225
Joystick (or game peripheral) ☐ 226
Modem ☐ 227
Monitor ☐ 228
Printer ☐ 229
Other (please specify) _____ 230

6. Which of the following peripherals do you plan to purchase in the next 12 months? (check all that apply)
Disk Drive ☐ 231
Joystick (or game peripheral) ☐ 232
Modem ☐ 233
Monitor ☐ 234
Printer ☐ 235
Other (please specify) _____ 236

7. Approximately how much do you plan to spend on your computer or computer-related items in the next 12 months?
Under $100 ☐ 237
100–499 ☐ 238
500–999 ☐ 239
1000–1999 ☐ 240
2000 or over ☐ 241
Don't know ☐ 242

8. Please put an X in the box which indicates your age group.
Under 18 ☐ 243
18–29 ☐ 244
30–44 ☐ 245
45–65 ☐ 246
Over 65 ☐ 247

9. What is the highest level you completed in school?
Some high school or less ☐ 248
High school graduate ☐ 249
Attended college 1–3 years ☐ 250
College graduate ☐ 251
Post-graduate study without degree ☐ 252
Master's degree ☐ 253
Doctoral degree ☐ 254

10. Would you describe your present occupation as
Student ☐ 255
Manufacturing/service ☐ 256
Clerical/technical ☐ 257
Sales ☐ 258
Educator ☐ 259
Professional ☐ 260
Other ☐ 261

11. What is the total combined annual income before taxes for all members of your household?
Less than $10,000 ☐ 262
10,000 to 14,999 ☐ 263
15,000 to 19,999 ☐ 264
20,000 to 24,999 ☐ 265
25,000 to 34,999 ☐ 266
35,000 to 49,999 ☐ 267
50,000 to 74,999 ☐ 268
75,000 or over ☐ 269

12. Please circle number 270 if you would like to receive information on other COMPUTE! publications.

COMPUTE!'s Best of Apple Reader Service
P.O. Box 2141
Radnor, PA 19089

Place Stamp Here

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 7478 DES MOINES, IOWA

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE!'s Apple
PO BOX 10954
DES MOINES, IOWA 50347

The first graphics program for the IIGS that makes all the Apples shine!

Apple IIGS, IIe, IIc

NEW

816/Paint™

Spark your creativity with an extraordinary color palette...and the graphic tools to put it to work.

Many color and black & white printers supported. Apple ImageWriters, Epsons and more.

★ ★ ★

Only 512K required for IIGS.

BAUDVILLE™

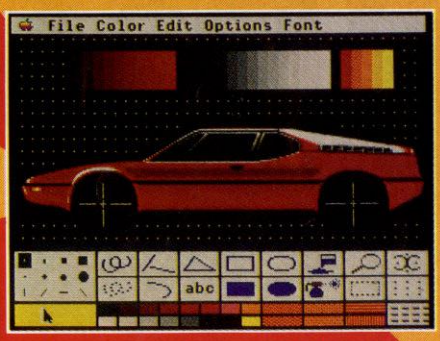
Introducing 816/Paint.

Take your Apple II to its limit. *816/Paint* is the first graphics program that works with all Apple Hi-Res graphics modes, including the 4096 color Super Hi-Res mode on the IIGS.

All you need is an Apple IIe, IIc, or IIGS. *816/Paint* gives you the tools you need for outstanding results, whether it's illustrations for video slides, clip art for desktop publishing, or just some creative fun.

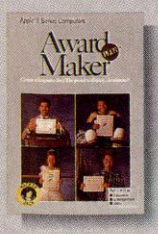
816/Paint includes two ProDOS disks: a 3½" 800K disk for the IIGS, and a 5¼" disk for IIe and IIc machines.

No matter which Apple II you own, *816/Paint* will deliver unsurpassed color and creative capabilities. $75.00



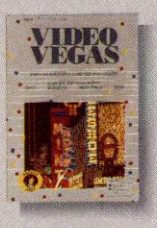
• Scale, rotate, stretch or paint • Full-screen editing • Color cycle animation • Prints to ImageWriter II or LaserWriter

Four more ways to make your Apple II shine.



Award Maker Plus
Choice of intricate colored borders, illustrations and personalized messages. $39.95

French parchment paper available at $12.95 for 100 sheets.



Video Vegas
Enjoy casino action at home:

- Blackjack
- Draw Poker
- Slots
- Keno

Apple II series, C64, IBM Atari XE/XL, Amiga, IBM, Atari ST

Take 1, Deluxe Version
Let your creativity soar with this video animation studio. Winner, 1986 Classroom Computer Learning Software Award. Includes free Animation Library. $59.95

Put the power of *Take 1* graphics and animation in your own programs. Try the *Take 1 Programmer's TOOLKIT.* Easy enough for the novice BASIC programmer, yet powerful enough for the most serious applications. $60.00

Ask your dealer for these Baudville programs or call us direct at 616-957-3036. VISA and MasterCard accepted for phone orders. Call or write for our free catalog today!

☎ 616-957-3036

Circle 159 on Reader Service Card.

BAUDVILLE™

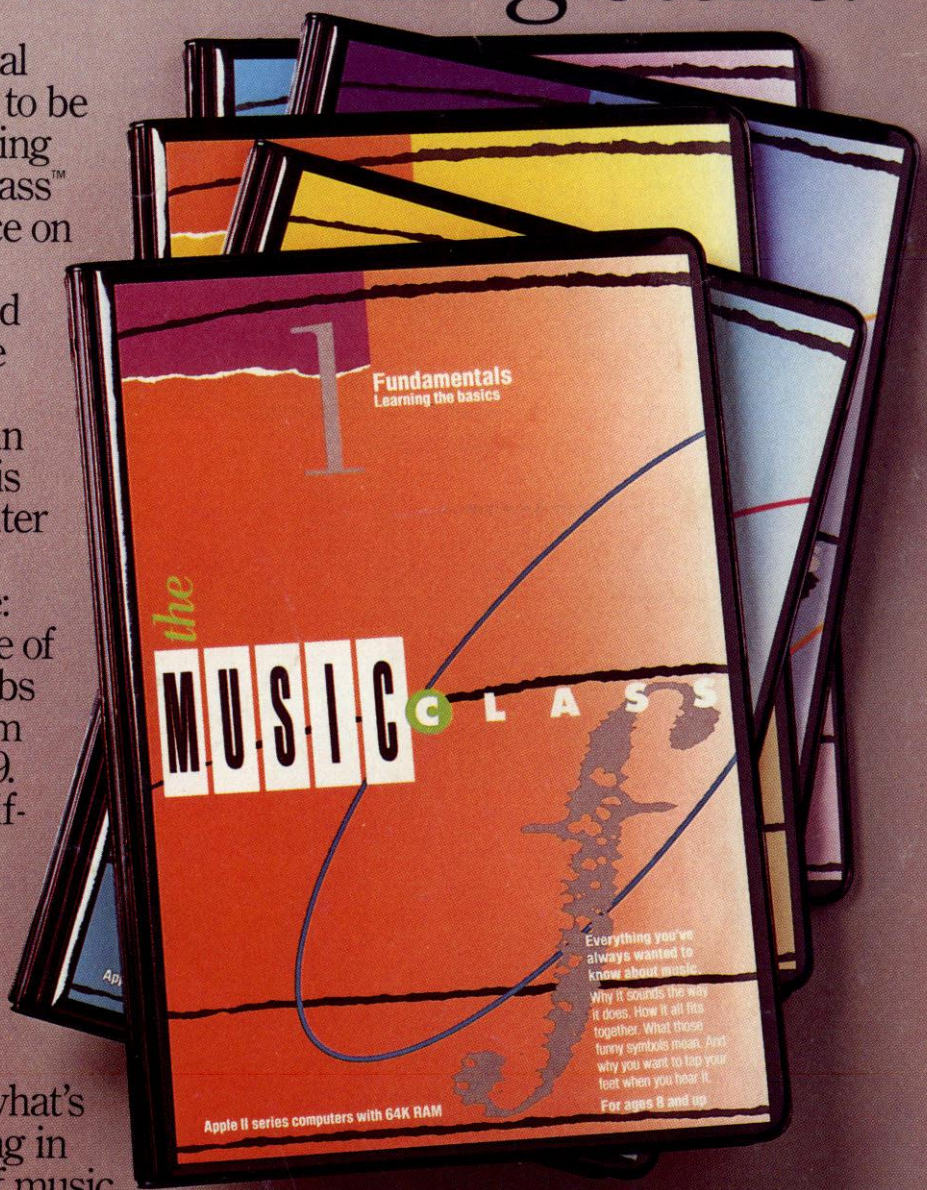
1001 Medical Park Drive, S.E., Grand Rapids, MI 49506

Presenting One Musical Group That Will Never Make The Cover Of Rolling Stone.

While most musical groups would love to be on the cover of Rolling Stone, The Music Class™ will settle for an appearance on your Apple computer.

Its playful graphics and challenging tests will guide you through the principles of music. All you need to run The Music Class software is an Apple II™ or IIGS™ computer with 64K memory.

The five programs are:
Fundamentals: Make sense of those skinny lines with blobs and tails. All the basics from rhythm to note reading. $49.
Rhythm: What the exact difference is between waltz, polka, jazz, rock, ragtime, and more. A comical guy by the name of Mr. Metro Gnome is the teacher. $49.
Ear Training: How to hear exactly what's happening in a piece of music. You'll never listen to The Rolling Stones quite the same way again. $49.
Music Symbols: No, they aren't called squiggles, blips, slashes and dots. You'll learn up to eighty musical symbols. $39.
Note Reading: Know what the differences are between an E-Flat eighth note and a B-Flat quarter note. This is where you'll learn to read music. $39. So order by calling toll free 1-800-843-1337. Or collect 612-854-9554. Ask about our Coda Catalog with just over 600 music software products. And get your music career rolling.



The Music Class allows kids to learn at their own speed. Whether they're beginners or child prodigies.

Wenger Music Learning Division, 1401 E. 79th Street, Minneapolis, Minnesota 55420-1590